UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

MATHEUS DE OLIVEIRA SILVA

DESENVOLVIMENTO DE UMA APLICAÇÃO GRÁFICA PARA RESOLUÇÃO DE PROBLEMAS DE ROTEAMENTO DE VEÍCULOS

João Pessoa
2024

MATHEUS DE OLIVEIRA SILVA

DESENVOLVIMENTO DE UMA APLICAÇÃO GRÁFICA PARA RESOLUÇÃO DE PROBLEMAS DE ROTEAMENTO DE VEÍCULOS

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Produção da UNIVERSIDADE FEDERAL DA PARAÍBA, como requisito parcial para a Obtenção do grau de Bacharel em Engenharia de Produção.

Orientador: Prof. Dr. Luciano Carlos Azevedo da Costa

João Pessoa

2024

Catalogação na publicação
Seção de Catalogação e Classificação

S586d Silva, Matheus de Oliveira.
Desenvolvimento de uma aplicação gráfica para resolução de problemas de roteamento de veículos / Matheus de Oliveira Silva. - João Pessoa, 2024.
66 f. : il.

Orientação: Luciano Carlos Azevedo da Costa Costa.
TCC (Graduação) - UFPB/CT.

1. Roteamento de veículos. 2. Janelas de Tempo. 3. Pesquisa Operacional. 4. Aplicação Gráfica. I. Costa, Luciano Carlos Azevedo da Costa. II. Título.

UFPB/BSCT CDU 658.5(043.2)

Elaborado por ROSANGELA GONCALVES PALMEIRA - CRB-216

MATHEUS DE OLIVEIRA SILVA

DESENVOLVIMENTO DE UMA APLICAÇÃO GRÁFICA PARA RESOLUÇÃO DE PROBLEMAS DE ROTEAMENTO DE VEÍCULOS

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Produção da UNIVERSIDADE FEDERAL DA PARAÍBA, como requisito parcial para a Obtenção do grau de Bacharel em Engenharia de Produção.

João Pessoa, 25 de outubro de 2024

BANCA EXAMINADORA

Documento assinado digitalmente
gov.br
LUCIANO CARLOS AZEVEDO DA COSTA
Data: 29/10/2024 10:06:34-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Luciano Carlos Azevedo da Costa
Universidade Federal da Paraíba

Documento assinado digitalmente
gov.br
ALESSANDRA BERENGUER DE MORAES
Data: 25/10/2024 16:52:20-0300
Verifique em https://validar.iti.gov.br

Profa. Alessandra Berenguer de Moraes
Universidade Federal da Paraíba

Documento assinado digitalmente
gov.br
HUGO HARRY FREDERICO RIBEIRO KRAMER
Data: 28/10/2024 20:48:27-0300
Verifique em https://validar.iti.gov.br

Prof. Hugo Harry Frederico Ribeiro Kramer
Universidade Federal da Paraíba

Dedico este Trabalho aos meus pais, pelo apoio constante e incentivo em todos os momentos desta jornada.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais pelo apoio incondicional em todos os momentos desta jornada acadêmica. Sem o incentivo e a confiança que sempre depositaram em mim, este trabalho não seria possível.

Ao professor e orientador Luciano Costa, expresso minha profunda gratidão pela paciência, pelos incentivos constantes e por acreditar na minha capacidade de desenvolver este trabalho. Sua orientação foi fundamental para o meu crescimento acadêmico e profissional.

Aos meus professores da graduação, agradeço por todo o conhecimento compartilhado e pelo apoio durante esses anos. Em especial, à professora Alessandra, cujo incentivo me motivou a continuar estudando Pesquisa Operacional, área pela qual desenvolvi grande apreço.

Aos meus amigos e colegas de graduação, que tornaram esta caminhada mais leve e significativa. Um agradecimento especial a Antônio Edson, Matheus Teixeira e Iordan Lucas pela amizade, companheirismo e por compartilharem comigo inúmeros momentos de aprendizado.

“Grandes realizações não são feitas por impulso, mas por uma soma de pequenas realizações.” (Vincent Van Gogh)

RESUMO

O presente trabalho tem como objetivo o desenvolvimento de uma aplicação gráfica para a resolução de problemas de roteamento de veículos, com foco no Problema de Roteamento de Veículos com Janelas de Tempo (PRVJT). Este problema é comumente enfrentado pelas empresas que buscam atender clientes dispersos geograficamente, considerando as restrições de capacidade dos veículos e respeitando os horários de agendamento dos clientes.

A aplicação proposta utiliza um modelo matemático para a resolução do problema, visando à minimização dos custos operacionais. Ela oferece uma interface intuitiva, permitindo ao usuário definir os pontos de entrega manualmente ou por meio do *upload* de planilhas eletrônicas no formato .xlsx. Além disso, a aplicação se integra a APIs externas, como o *OpenStreetMap* e o *Open Source Routing Machine*, para o cálculo de distâncias, geometria das rotas e visualização do mapa, fornecendo uma solução completa e automatizada para o Problema de Roteamento de Veículos.

Palavras-chave: Roteamento de veículos, Janelas de Tempo, Pesquisa Operacional, Aplicação Gráfica

ABSTRACT

The present work aims to develop a graphical application for solving vehicle routing problems, focusing on the Vehicle Routing Problem with Time Windows (VRPTW). This problem is commonly faced by companies seeking to serve geographically dispersed customers, considering vehicle capacity constraints and adhering to customers' scheduled appointment times. The proposed application relies on a mathematical model to solve the problem, aiming to minimize operational costs. It offers an intuitive interface, allowing users to define delivery points manually or by uploading spreadsheets in .xlsx format. Additionally, the application integrates with external APIs, such as OpenStreetMap and the Open-Source Routing Machine, for calculating distances, route geometry, and map visualization, providing a complete and automated solution for the Vehicle Routing Problem.

Keywords: Vehicle Routing, Time Windows, Operational Research, Graphical Application.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

1.1 APRESENTAÇÃO DO TEMA

A constante evolução tecnológica trouxe mudanças significativas em diversos setores da economia, especialmente com o advento da Indústria 4.0. Esse conceito, também chamado de Quarta Revolução Industrial, integra tecnologias avançadas como a Internet das Coisas, *Big Data*, Inteligência Artificial e automação (Büchi; Cugno; Castagnoli, 2020). Essas inovações possibilitam o desenvolvimento de soluções que automatizam processos e otimizam a tomada de decisões em problemas complexos de engenharia (Pereira; Romero, 2017).

A Indústria 4.0 revolucionou não apenas as indústrias, mas também a logística e toda a cadeia de suprimentos, gerando um ambiente onde as decisões podem ser tomadas de maneira mais dinâmica e precisa. Essa revolução envolve a utilização de dados em tempo real, a conectividade entre dispositivos e a automação de tarefas, resultando em maior eficiência, redução de custos e melhorias na qualidade do serviço prestado (Culot; Nassimbeni; Orzes; Sartor, 2020). Isso resulta em uma redução de custos operacionais e em ganhos significativos de competitividade para as empresas.

Entre os problemas enfrentados pelas empresas nesse cenário de avanço tecnológico, destacam-se os Problemas de Roteamento de Veículos (PRVs). O PRV tem como objetivo encontrar as melhores rotas para uma frota de veículos que precisa atender clientes dispersos geograficamente, minimizando os custos e respeitando restrições, como a capacidade dos veículos. Uma variante bastante importante deste problema, conhecida como Problema de Roteamento de Veículos com Janelas de Tempo (PRVJT), que inclui restrições de janelas de tempo que os veículos devem respeitar ao atender os clientes, tem recebido grande atenção da academia e da indústria, devido ao seu impacto direto na eficiência logística (Zhang; Ge; Yang; Tong, 2021).

Além disso, a utilização de aplicações gráficas desempenha um papel fundamental na implementação de soluções para problemas complexos, como o PRVJT. Ferramentas que oferecem interfaces intuitivas e de fácil utilização permitem que os usuários interajam de maneira mais eficiente com os sistemas, facilitando a

inserção de dados, a visualização dos resultados e a interpretação das rotas geradas (Bodin, 1990). Isso reduz a resistência dos usuários em utilizar aplicações computacionais, sendo a facilidade de uso um fator crucial para o sucesso de qualquer aplicação.

1.2 JUSTIFICATIVA

Com a crescente demanda por serviços logísticos mais ágeis e eficientes, a otimização do roteamento de frotas de veículos tornou-se uma necessidade para empresas que buscam competitividade. A automação e o uso de ferramentas computacionais são essenciais para que as empresas alcancem a eficiência operacional necessária. A redução de custos é a principal prioridade para a gestão de logística no Brasil. Segundo pesquisa realizada pela NSClub, 58% dos executivos e executivas apontaram ações voltadas para a redução de custos logísticos, destacando-se a implementação de tecnologias nos processos como uma estratégia relevante, mencionada por 57% dos entrevistados (PRESA, 2023). A utilização de algoritmos de otimização para resolver o PRVs, além de reduzir custos, também melhora a eficiência das operações e melhora o aproveitamento das rotas, resultando em menos veículos nas ruas e menores emissões de gases poluentes (Novaes, 2007).

Sistemas de roteamento pagos, são amplamente utilizados tanto por empresas brasileiras quanto por empresas no exterior. No entanto, esses sistemas apresentam um alto custo de aquisição e manutenção, tornando-se inviáveis para pequenas e médias empresas. Além disso, esses softwares oferecem pouca flexibilidade para personalizações específicas que atendam às necessidades particulares de cada empresa (Melo; Filho, 2001).

Este trabalho insere-se em um contexto que busca desenvolver uma aplicação gráfica que integra diferentes tecnologias para resolver o PRV de maneira eficaz e acessível, auxiliando as empresas na tomada de decisões.

1.3 OBJETIVO GERAL

Desenvolver uma aplicação gráfica para a resolução de Problemas de Roteamento de Veículos (PRV), proporcionando uma interface intuitiva e de fácil utilização, que permita que os usuários configurem e visualizem rotas de maneira simples e eficiente

1.4 OBJETIVOS ESPECÍFICOS

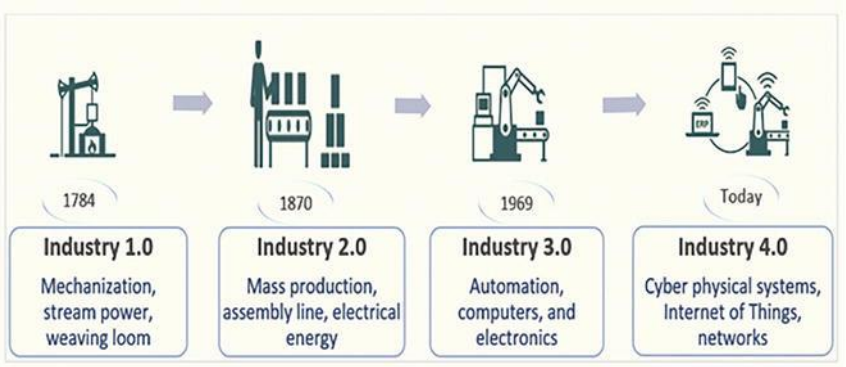
- Desenvolver uma interface gráfica que permita a inserção manual dos dados dos clientes ou o *upload* de planilhas eletrônicas (.xlsx);
- Integrar a aplicação com *APIs* externas, como *OpenStreetMap* e *Open Source Routing Machine*, para cálculo das distâncias entre os pontos, visualização geográfica e geocodificação dos pontos inseridos no mapa;
- Implementar modelos para a resolução do PRVJT: para fins de aplicabilidade, utilizar o modelo clássico do Problema de Roteamento de veículos com Janelas de Tempo;
- Permitir a personalização dos parâmetros de entrada, oferecendo aos usuários a possibilidade de configurar aspectos essenciais, tais como o número de veículos, a capacidade de carga, os horários de saída, as demandas dos clientes, as janelas de tempo e a localização do depósito

2 FUNDAMENTAÇÃO TÉORICA

2.1 INDÚSTRIA 4.0

Também conhecida como a Quarta Revolução Industrial, essa transformação está sendo implementada atualmente e se refere, principalmente, ao conceito de fábricas onde os equipamentos utilizam Internet das Coisas, impressão 3D, Inteligência Artificial, aprendizado de máquina, *big data* e realidade aumentada para tomar decisões sem intervenção humana (Büchi; Cugno; Castagnoli, 2020). Trata-se de um sistema interligado, resultando em sistemas de produção *ciberfísicos*, o que dá origem às fábricas inteligentes. Nelas, os sistemas de produção, componentes e pessoas interagem por meio de uma rede quase autônoma. Um exemplo disso são máquinas que podem prever falhas e iniciar operações de manutenção por conta própria, ou uma logística que se auto-organiza para lidar com mudanças inesperadas na produção (Culot; Nassimbeni; Orzes; Sartor, 2020).

Figura 1 - Ilustração das revoluções industriais ao longo do tempo



Fonte: (Abdelmajied, 2022)

A Figura 1 ilustra as variações das revoluções industriais ao longo do tempo. Na Indústria 1.0, começa-se a trabalhar com sistemas mecanizados, como teares de tecelagem e o aproveitamento do córrego do rio para girar moinhos. Em seguida, na Indústria 2.0, surge o conceito de produção em massa, com a criação da linha de montagem e o uso da energia elétrica. Posteriormente, na Indústria 3.0, se cria a automação de tarefas em conjunto com a computação e a eletrônica. Hoje se vive na Indústria 4.0, a qual está desenvolvendo fábricas inteligentes com a internet das

coisas, aprendizado de máquina e inteligência artificial. Nos últimos anos, as organizações têm mudado os seus modelos de negócio de forma rápida, em consequência do surgimento de tecnologias disruptivas e o desenvolvimento da Indústria 4.0, alterando a forma como os produtos e serviços são vendidos e entregues (Karnik; Bora; Bhadri; Kadambi; Dhatrak, 2022).

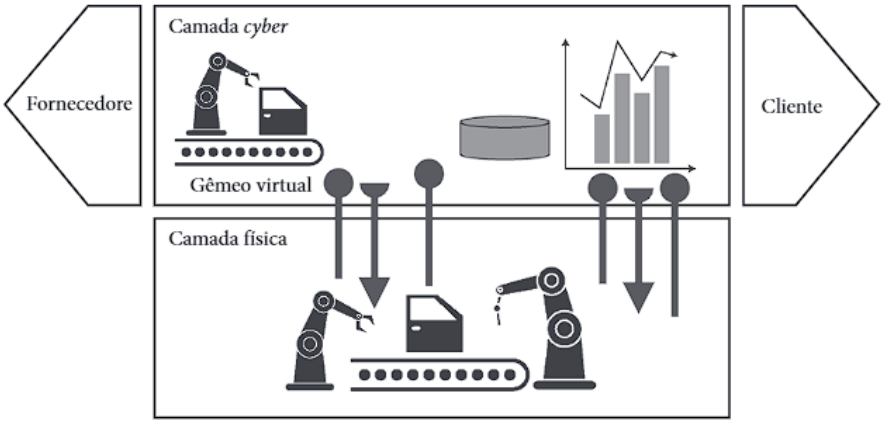
2.1.1 COMPONENTES DA INDÚSTRIA 4.0

Esta seção fornece uma visão geral dos componentes da indústria 4.0, os quais se interligam em um sistema de valor agregado. A emergente tecnologia da indústria 4.0 está moldada em grande parte pela integração técnica de sistemas ciberfísicos nos processos de fabricação, além do uso da internet das coisas e serviços industriais (Pereira; Romero, 2017)

2.1.2 SISTEMAS CIBERFÍSICOS

É a combinação de processos que interligam a computação e processos físicos, a fim de monitorar e controlar entidades físicas em tempo real, além de simular esses processos, a partir do ambiente virtual. A Figura 2 mostra, de forma simples, como é a estrutura desse sistema. A camada física é tudo que seja físico no chão de fábrica, como: máquinas operatrizes, robôs, fornos, caldeiras, esteiras etc.

Figura 2 - Arquitetura de sistema ciberfísicos



Fonte: (Petroni; Gloria Junior; Gonçalves, 2018)

A camada cyber é formada pelas aplicações de TI, como uma cópia virtual do ambiente físico, interfaces de gerenciamento e sistemas para captura e armazenamento de dados em grande quantidade. Esses sistemas interagem por meio de sensores e atuadores. Os sensores captam informações do sistema físico e as transformam em pulsos elétricos para o sistema cyber. Já os atuadores realizam intervenções no mundo físico a partir de sinais e comandos digitais (Petroni; Gloria Junior; Gonçalves, 2018).

2.1.3 SISTEMAS EM NUVEM

A computação em nuvem é um modelo de rede que permite o compartilhamento de recursos entre os usuários. Esse acesso é configurável e pode ser rapidamente disponibilizado conforme solicitado, com o mínimo esforço por parte do usuário, demandando a mínima interação com o provedor de serviços (Paz; Loos, 2020). Existem diversas vantagens na adoção desse modelo na Indústria 4.0, pois ele permite uma mínima interação com o servidor, ampliando as possibilidades de automação. Segundo relatório da Foresight (2023), os benefícios da computação em nuvem são inúmeros, destacando-se os seguintes:

- Flexibilidade e fácil compartilhamento de dados: manipulação simplificada dos arquivos na nuvem, permitindo que várias pessoas os acessem e manipulem simultaneamente;
- Monitoramento de mobilidade: proporciona rastreabilidade das ações e maior controle das operações;
- Economia: aumento da rentabilidade proporcionado pelo rápido acesso às informações, além do melhor controle e rastreabilidade dos dados.

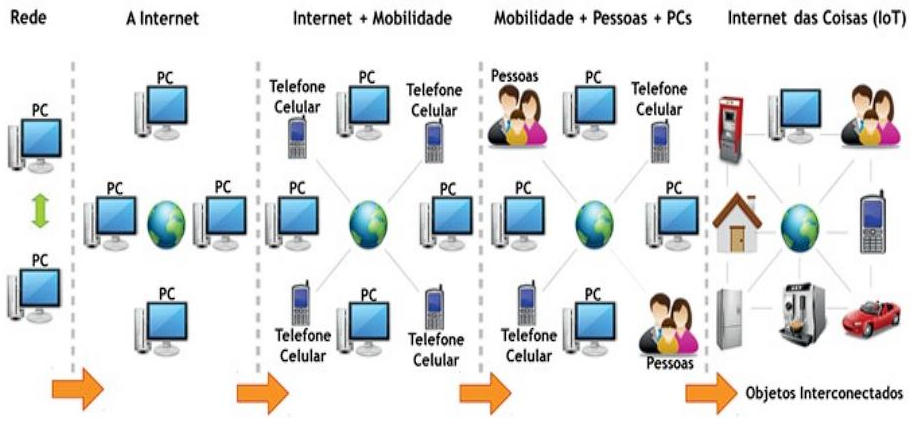
2.1.4 INTERNET DAS COISAS

Também conhecida como *Internet Of Things* (IoT) é um conceito que surgiu recentemente com o avanço da tecnologia, combinando diversas dessas inovações, se baseando na interação entre objetos físicos e a internet. Ela pode ser definida como uma comunicação *Machine to Machine* (M2M), em que diferentes dispositivos compartilham informações e dados para executar tarefas específicas.

Com o uso de sensores e dispositivos, a IoT possibilita a comunicação entre objetos e requer um sistema computacional para gerenciar o comportamento de cada “coisa” conectada a uma rede (Paz; Loos, 2020).

A Figura 3 ilustra a evolução até a chegada da *IoT*, destacando a interconexão entre os dispositivos.

Figura 3 - evolução da internet até chegar no IoT



Fonte: Adaptado de Perera; Liu; Jayawardena; Chen, 2014.

A IoT pode ser aplicada em diversos setores do cotidiano. Segundo Paz e Loos (2020), alguns exemplos de destaque são:

- Casa: Sistema de monitoramento de temperatura que, ao detectar uma elevação em relação à temperatura preferida do usuário, ajusta automaticamente o ambiente para mantê-lo confortável.
- Automóvel e Logística: Automóveis capazes de estacionar de forma autônoma, sem intervenção humana, e protótipos de veículos autodirigíveis. No setor logístico, há sensores que monitoram o estado do veículo, sugerindo manutenções preventivas e corretivas.
- Cidades Inteligentes: Integração de toda a infraestrutura urbana, como lixeiras, vagas de estacionamento, sensores de alerta contra inundações, frota pública, entre outros. Quanto mais objetos conectados, mais eficiente será a administração pública, resultando em uma melhor qualidade de vida para os cidadãos.

2.1.5 BIG DATA

Com o aumento exponencial de dados devido ao advento da internet e à proliferação de dispositivos como celulares e computadores, surgiu o conceito de *Big Data*. Segundo Galdino (2016), esse grande volume de dados possui cinco características principais, conhecidas como os 5 Vs do *Big Data*:

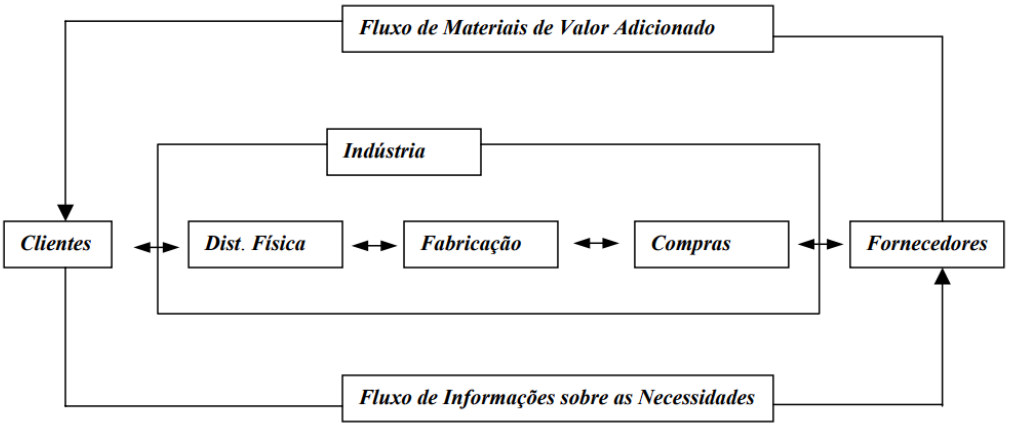
- Volume: quantidade de dados acumulados;
- Variedade: diferentes formas de extração, propagação e tipos de dados;
- Velocidade: taxa de transmissão dos dados;
- Veracidade: confiabilidade dos dados;
- Valor: insights e resultados gerados a partir desses dados.

Esses dados podem ser classificados, segundo Galdino (2016), em três categorias: dados estruturados, que são organizados em formato tabular, onde cada linha e coluna possuem um relacionamento entre si, geralmente gerenciado por um Sistema de Gerenciamento de Banco de Dados (SGBD); dados semiestruturados, que apresentam uma estrutura parcial, sendo irregulares ou incompletos, como documentos *HTML* e *logs* de *websites*; e dados não estruturados, que não possuem qualquer estrutura prévia e não podem ser agrupados em tabelas, como vídeos, imagens e e-mails.

2.2 LOGÍSTICA

Segundo Novaes (2007), o Council of Logistics Management (1996) define a logística como um processo de planejar, implementar e controlar de maneira eficiente o fluxo e o armazenamento de matérias-primas, desde o ponto de origem até o ponto de consumo, com o objetivo de atender às exigências dos clientes. A Figura 4 demonstra que a logística não se restringe apenas ao ambiente fabril ou ao fluxo de materiais, mas também abrange o fluxo de informações e se estende além das fronteiras da indústria, incluindo clientes e fornecedores.

Figura 4 - Processo de Gerenciamento Logístico



Fonte: Christopher (1998)

De acordo com Novaes (2007), a distribuição física é composta por componentes físicos e informacionais, tais como:

- Instalações físicas (centros de distribuição e armazéns): são espaços destinados ao armazenamento de produtos, de onde podem ser transferidos para as lojas ou diretamente para os clientes. É fundamental que essas instalações estejam localizadas em regiões com fácil acesso para carga e descarga.
- Estoque de produtos: representa o custo de capital relacionado ao volume de produtos armazenados em diferentes locais, como depósitos de fábrica, centros de distribuição, lojas de varejo e veículos de transporte.
- Veículos: responsáveis pelo transporte das mercadorias até os pontos de consumo, considerando que os centros consumidores estão em locais diversos. Veículos de maior porte são utilizados para abastecer as fábricas e os centros de distribuição, enquanto veículos menores são empregados para a entrega aos centros consumidores.
- Informações diversas: para operar e controlar um sistema de distribuição eficiente, é necessário dispor de informações variadas, como localizações geográficas, sistemas de roteamento de veículos, quantidades de produtos a serem entregues a cada cliente e capacidade dos veículos.
- Hardware e software diversos: grande parte das atividades de distribuição é planejada, programada e controlada por meio de aplicativos que auxiliam na

roteirização de veículos, controle de pedidos, monitoramento e outras tarefas. Esses aplicativos (*softwares*) funcionam com o suporte de computadores e dispositivos *(hardwares*).

- Custos: para se manter competitivo, é necessário possuir uma estrutura de custos adequada e constantemente atualizada.
- Pessoal: com o avanço da tecnologia, é essencial que os profissionais acompanhem essas mudanças para aproveitar ao máximo as novas ferramentas. Quando necessário, deve-se investir na capacitação e atualização do capital humano.

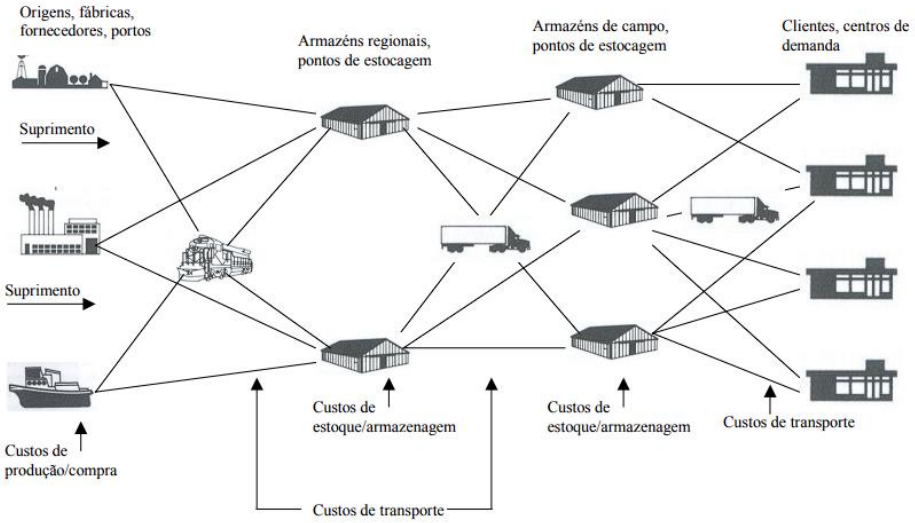
2.2.1 CUSTO LOGÍSTICO

Segundo Ballou (1995), o custo logístico total é composto pela soma dos custos de transporte, armazenagem e processamento de pedidos. No planejamento de uma rede de distribuição logística, é fundamental definir quais instalações devem ser utilizadas, quantas devem existir, onde devem ser localizadas, quais produtos e clientes devem ser designados a elas, quais serviços de transporte devem ser empregados entre as instalações e como essas instalações devem ser atendidas (Ballou, 2001).

A Figura 5 ilustra uma rede de distribuição genérica com um fluxo simplificado de produtos, evidenciando a complexidade de planejar uma distribuição logística em rede. É essencial considerar dois aspectos principais: o aspecto espacial e o temporal. O aspecto espacial (geográfico) refere-se à localização das instalações, como armazéns, plantas e lojas de varejo. Para determinar o número, o tamanho e a localização dessas instalações, é necessário equilibrar os requisitos de serviço ao cliente com os custos de produção/compra, manutenção de estoques, instalações e transporte. O aspecto temporal está relacionado à disponibilidade do produto para atender às metas de serviço ao cliente, seja por meio de um baixo tempo de reabastecimento ou pela manutenção de um estoque próximo aos clientes. Para orientar a gestão desses aspectos, existem três estratégias importantes: minimizar todos os custos logísticos relevantes, respeitando as restrições de serviço logístico dos clientes; maximizar o nível de serviço logístico ao cliente, garantindo que as expectativas e necessidades sejam atendidas de maneira eficiente; e maximizar a

contribuição do lucro gerada pela logística, otimizando a margem resultante da receita proporcionada pelo nível de serviço logístico ao cliente e os custos associados à sua prestação (Ballou, 2001).

Figura 5 - Rede Logística Genérica

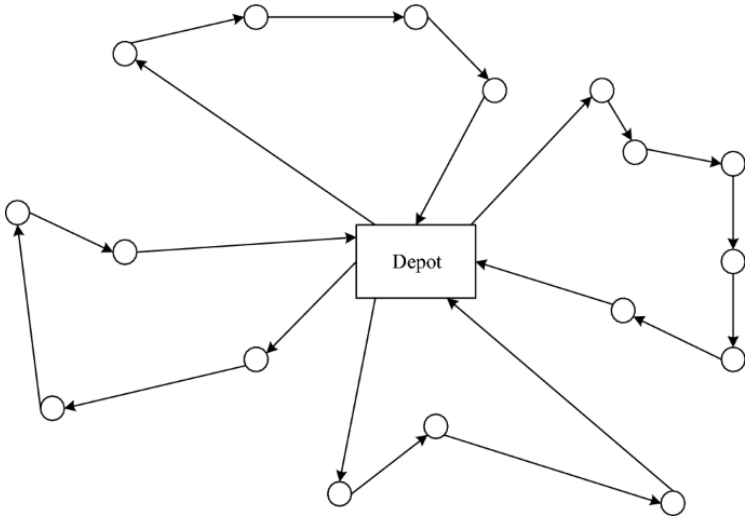


Fonte: (Ballou, 2001)

2.2.2 ROTEAMENTO DE VEÍCULOS

Também chamado por roteirização de veículos, é o processo de determinação um ou mais roteiros ou sequências de paradas a serem cumpridos por veículos de uma frota, com o objetivo de visitar um conjunto de pontos geográficos pré-determinados, que necessitam de atendimento (Cunha,1997). O primeiro problema de roteirização a ser estudado foi o problema chamado de caixeiro viajante, que se consiste em encontrar um roteiro ou sequência de cidades a serem visitadas por um caixeiro viajante que minimize a distância total percorrida e que as cidades sejam visitas apenas uma vez (Applegate *et al*., 2007). A partir disso, novas restrições vêm sendo estudadas e incorporadas ao problema do caixeiro viajante, visando representar de forma mais precisa os desafios que envolvem o roteamento de veículos. Entre essas restrições estão: horários de atendimento, capacidades dos veículos, frotas compostas por veículos de diferentes tamanhos, duração máxima dos roteiros dos veículos, entre outras (Cunha, 2000).

Figura 6 - Diagrama de um problema de Roteamento de Veículo



Fonte: (Zhang; Ge; Yang; Tong, 2021).

Portanto, os problemas de roteamento são desdobramentos do clássico problema do caixeiro viajante, porém com mais restrições e múltiplos caixeiros. O problema de roteamento de veículos foi proposto pela primeira vez por Dantzig e Ramser (1959) e foi aplicado na otimização do roteamento de petroleiros entre a refinaria de Atlanta e seus postos de gasolina. Cada cliente possui uma demanda específica, o depósito é responsável por fornecer as mercadorias, e uma frota realiza a distribuição. O objetivo é atender às necessidades dos clientes utilizando a rota mais curta e com o menor tempo possível, respeitando algumas restrições. De maneira geral, o problema de roteamento de veículos pode ser descrito da seguinte forma: sob determinadas condições de restrição, busca-se a rota ideal para realizar a entrega de um ou mais pontos iniciais para vários clientes localizados em diferentes lugares, conforme ilustrado na Figura 6 (Zhang *et al.*, 2021)

Dessa forma, o objetivo é projetar uma rota com custo mínimo, onde cada cliente é visitado apenas uma vez por um veículo, e todos os veículos partem de um ponto de origem e retornam a ele após a conclusão das entregas. O problema deve satisfazer algumas restrições, sendo as mais comuns: capacidade dos veículos e janelas de tempo. Conforme Zhang *et al*. (2021) o problema de roteamento de veículos combina diversos fatores, como:

- Estrada: representada por uma rede que interliga as rotas que passam por cada cliente até retornar ao depósito;

- Ponto do cliente: representa o local de atendimento do serviço;
- Depósito: ponto de partida e chegada dos veículos;
- Veículo: responsável pela distribuição até o ponto do cliente e pelo retorno ao depósito;
- Requisitos de organização de transporte: a rota dos veículos geralmente depende da natureza das mercadorias transportadas, das características dos clientes e dos próprios veículos.

Resolver esse tipo de problema é computacionalmente caro, categorizado como NP-difícil, pois os problemas do mundo real envolvem restrições complexas, como o número de soluções crescendo de forma exponencial com o aumento de cada cliente. Nos últimos anos a velocidade de processamento e a capacidade dos computadores vêm aumentado, permitindo a resolução de problema cada vez mais complexos e aplicações generalizadas de cenários de distribuição logística (Lenstra; Kan, 1981).

Segundo Pessoa *et al.* (2020), problemas com 100 clientes, que antes levavam horas para serem resolvidos, agora podem ser solucionados em menos de 1 minuto. Isso significa que muitas outras instâncias do mundo real agora se tornam viáveis para resolução. Além disso, Pecin *et al.* (2017) demostra no seu trabalho que já existem algoritmos capazes de resolver instanciais de 200 clientes.

De acordo com Liu *et al*. (2023), os algoritmos exatos são aqueles que garantem a solução ideal para um problema de PRV. Esses métodos incluem técnicas como pesquisa direta em árvore e programação dinâmica. No entanto, também existem soluções heurísticas, que são soluções aproximadas das ótimas, permitindo uma resolução mais rápida. As heurísticas são aplicadas em casos em que os algoritmos exatos são computacionalmente caros, e incluem técnicas como heurística construtiva, algoritmo de poupança e o algoritmo de Clarke e Wright. Além disso, existem os métodos metaheurísticos, que consistem em algoritmos projetados para explorar um vasto espaço de soluções em busca de uma solução de boa qualidade. Esses métodos geralmente combinam várias técnicas heurísticas para criar estratégias de pesquisa mais sofisticadas. Entre os métodos metaheurísticos estão o recozimento simulado, busca Tabu, algoritmo genético, otimização por colônias de formigas e otimização por enxame de partículas. Por fim, há os métodos híbridos, que combinam duas ou mais técnicas para formar uma estratégia de

pesquisa mais robusta. Eles geralmente combinam métodos exatos com heurísticas, ou heurísticas com metaheurísticas, para encontrar soluções de alta qualidade em um tempo razoável.

Com base no problema básico de roteamento de veículos (PRV), diversas extensões, adaptações e aplicações práticas foram desenvolvidas, de acordo com Zhang et al. (2021), elas incluem:

1. **Problema de roteamento de veículos capacitados**: adiciona a restrição de carga do veículo
2. **Problema de roteamento de veículos com janela de tempo**: Além da capacidade, adiciona a restrição de janela de tempo;
3. **Problema de roteamento de veículos de entrega dividida**: quando a demanda de entrega é maior que a capacidade dos veículos, utiliza-se a abordagem da entrega dividida;
4. **Problema de roteamento dinâmico de veículos**: demanda dinâmica e atualização do roteamento de acordo com informações de tráfego em tempo real;
5. **Problema de roteamento de veículos com demanda estocástica**: as demandas dos veículos não são conhecidas previamente, as demandas são tratadas como variáveis aleatórias com distribuição de probabilidade;
6. **Problema de roteamento de veículos com entrega-coleta simultânea**: quando se exige na mesma entrega serviços de entrega e coleta;
7. **Problema de roteamento de veículo aberto**: veículo aberto é tratado como um veículo alugado e não retorna ao deposito de origem;
8. **Problema de roteamento de veículos verdes**: utilização de veículos ecológicos, como: elétricos, híbridos ou movidos a combustíveis alternativos. Fazer o planejamento das rotas levando em conta a autonomia dos veículos verdes;
9. **Problema de roteamento de viagem de veículo**: possibilidade da inclusão onde um ou mais veículos realizam viagens específicas, considerando capacidades, janelas de tempo, tempos de serviço e outras condições operacionais.

2.2.3 ROTEAMENTO DE VEÍCULOS COM JANELAS DE TEMPO

Apesar de existirem muitas variantes de PRVs, este trabalho focará na variante com adição de janelas de tempo (PRVJT), que será detalhada a seguir.

O PRVTJ é o problema de determinar um conjunto de rotas de menor custo para uma frota de veículos homogêneos, denotados por V, um conjunto de clientes C e um grafo direcionado $G = (V, A)$. O grafo consiste em $|C| + 2$ vértices, onde os clientes são representados por {*1, 2, ..., n*} e o depósito é representado pelo vértice 0 e $n + 1$ como deposito de retorno. O conjunto de arcos do grafo, pode ser denotado por A, que representa as conexões entre o depósito e os clientes e entre os próprios clientes. Nenhum arco termina no vértice 0, e nenhum arco se origina no vértice $n + 1$. Para cada arco (i, j), onde $i \neq j$, associados a um custo c_{ij} e um tempo t_{ij}, que pode ser incluído no tempo de serviço do cliente i. Cada veículo tem uma capacidade Q e cada cliente i tem uma demanda d_i. Cada cliente i tem uma janela de tempo $[a_i, b_i]$. Um veículo deve chegar ao cliente antes de bi. O depósito também tem uma janela de tempo $[a_0, b_0]$. Os veículos não podem deixar o depósito antes de a_0 e devem retornar antes ou no momento b_{n+1}.No modelo, assume-se que os valores q, a_i, b_i, d_i, c_{ij} são inteiros não-negativos, enquanto os tempos t_{ij} são inteiros positivos. Além disso, pressupõe-se que tanto o custo c_{ij} quanto o tempo t_{ij} satisfazem a desigualdade triangular. O problema é modelado utilizando duas variáveis de decisão: x_{ijk} e s_{ik}. A variável x_{ijk} assume o valor 1 se, na solução ótima, o veículo k percorre o arco (i, j), e 0 caso contrário. Já s_{ik} denota o instante em que o veículo k começa a atender o cliente i. Se o veículo k não atender o cliente i, s_{ik} não terá significado.

O objetivo é projetar um conjunto de rotas de custo mínimo para os veículos, de modo que cada cliente seja visitado exatamente uma vez, todas as rotas comecem no vértice 0 e terminem no vértice $n + 1$, respeitando as restrições de capacidade e janelas de tempo. Dessa forma, o problema pode ser formalizado matematicamente como:

Modelo:

$$\min \sum_{k \in V} \sum_{i \in V} \sum_{j \in V} c_{ij}\, x_{ijk} \qquad (1)$$

Sujeito à:

$$\sum_{k \in V} \sum_{j \in N} x_{ijk} = 1 \qquad \forall i \in C \qquad (2)$$

$$\sum_{i \in C} d_i \sum_{j \in N} x_{ijk} \leq Q \qquad \forall k \in V \qquad (3)$$

$$\sum_{j \in N} x_{0jk} = 1 \qquad \forall k \in V \qquad (4)$$

$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0 \qquad \forall h \in C, \forall K \in V \qquad (5)$$

$$\sum_{i \in N} x_{i,n+1,k} = 1 \qquad \forall k \in V \qquad (6)$$

$$s_{ik} + t_{ij} - K\left(1 - x_{ijk}\right) \leq s_{jk} \qquad \forall i, j \in N, \quad \forall k \in V \qquad (7)$$

$$a_i \leq s_{ik} \leq b_i \qquad \forall i \in N, \quad \forall k \in V \qquad (8)$$

$$x_{ijk} \in \{0,1\} \qquad \forall i, j \in N, \quad \forall k \in V \qquad (9)$$

A função objetivo (1) busca minimizar o custo total de transporte. As restrições (2) garantem que cada cliente seja atendido apenas uma vez, enquanto a restrição (3) assegura que nenhum veículo transporte uma carga superior à sua capacidade. As restrições (4) - (6) garantem que cada veículo saia do depósito (0), visite um cliente e depois retorne ao depósito ($n + 1$). As restrições (7) calculam os horários de chegadas nos clientes. As restrições (8) garantem o cumprimento das janelas de tempo, e as restrições (9) definem o domínio das variáveis de decisão.

2.3 TECNOLOGIAS

2.3.1 ARQUITETURA MONOLÍTICA E MICROSSERVIÇOS

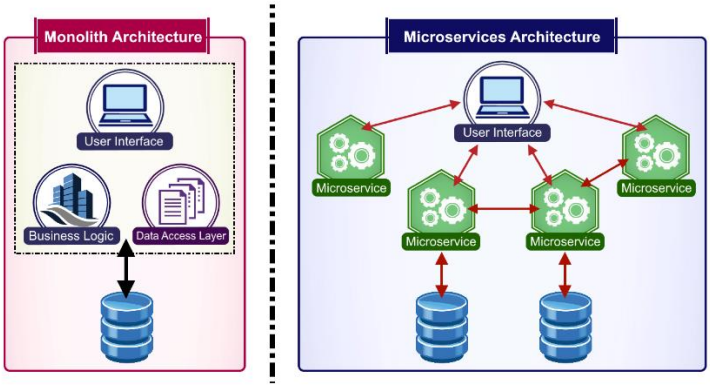
A arquitetura é composta por um conjunto de estruturas e elementos necessários para o funcionamento de um sistema. É essencial estudar os principais tipos de arquitetura para garantir que ela esteja alinhada aos objetivos de negócio da empresa (Bass; Clements; Kazman, 2015).

A arquitetura Monolítica, utiliza uma única tecnologia para todo o desenvolvimento da aplicação. Segundo Nielsen (2015), qualquer alteração

concluída nesse sistema, envolve a construção e implantação de uma nova versão de todo o sistema. Dentre as vantagens de adotar essa arquitetura está na estabilidade, grandes empresas utilizam esse sistema, como: IBM, Sun Microsystems e BMC, que contam com profissionais que possuem um alto nível de conhecimento sobre os seus produtos. Entre algumas desvantagens: são sistemas rígidos e difícil adaptabilidade a novas necessidades. Sua tecnologia é proprietária o que significa uma dependência entre o cliente e a empresa fornecedora. Os custos de aquisição, renovação e suporte são altos.

A arquitetura de microsserviços, conforme declarada pela AWS (2024), é uma abordagem arquitetônica para o desenvolvimento de aplicações em que a aplicação é composta por pequenos serviços independentes que se comunicam entre si por meio de *APIs*. Essa arquitetura permite a criação de aplicações grandes, complexas e escaláveis, compostas por processos menores e independentes, facilitando a comunicação e manutenção (Kratzke, 2017). A Figura 7 ilustra visualmente as diferenças entre a arquitetura de microsserviços e a monolítica.

Figura 7 - Representação das diferenças entre as duas arquiteturas



(Tapia; Mora; Fuertes; Aules; Flores; Toulkeridis, 2020)

2.3.2 FRONTEND, BACKEND, APIS E FRAMEWORK

Segundo Szwarcwald (2020), nos últimos anos as tecnologias de informação e comunicação vêm crescendo de forma intensa, resultando em um aumento significativo no número de *sites*. Geralmente, as aplicações são divididas em duas partes: a parte do usuário, denominada *frontend*, e a parte do servidor, chamada

backend. Para que essas aplicações funcionem, elas operam por meio de solicitações que os clientes (*frontend)* enviam ao servidor *(backend)*, geralmente por meio de métodos *HTTPS* (Noleto, 2022).

De acordo com Polidoro (2023), o *frontend* é a parte da aplicação voltada para o usuário, sendo responsável pela interação direta com ele. De maneira geral, o objetivo é traduzir o design e a interface da aplicação em código. Essa camada envolve o desenvolvimento utilizando linguagens como: *HTML*, *CSS* e *JavaScript*, entre outras tecnologias voltadas para o *frontend*.

- *HTML* (*Hypertext Markup Language*): define a estrutura da aplicação e o conteúdo da página web;
- *CSS* (*Cascading Style Sheets*): utilizado para o estilo, layout e aprimoramentos visuais;
- *JavaScript*: permite a interatividade e o comportamento dinâmico da aplicação

Segundo Noleto (2022), o *backend* envolve o processo de criação e manutenção do servidor da aplicação, incluindo a escrita de códigos para armazenamento, processamento e comunicação de dados com o *frontend*. As linguagens de programação comumente utilizadas são: *Python*, *Java* e *Ruby*.

Uma *API* (*Application Programming Interface*) lida com a integração entre diferentes aplicações e sistemas, automatizando processos manuais. Por exemplo, uma loja pode usar a *API* de um banco em seu site para processar pagamentos. Outro exemplo é a Netflix, que utiliza *APIs* de terceiros para exibir as imagens em seus catálogos (Arvind *et al.*, 2024)

De acordo com Sousa (2024) existem várias formas de comunicação entre esses dois sistemas, cada uma com seus benefícios e particularidades, no geral as mais utilizadas são: troca de dados por meio de requisições *HTTP* e utilização de *APIs*. Na troca de dados por meio de requisições *HTTP*, o *frontend* envia requisições ao servidor *backend*, por meio de métodos *HTTP*, como:

- *Método GET*: utilizado para solicitação de dados ao *backend;*
- *Método POST*: utilizado para enviar dados ao *backend;*
- *Método PUT*: utilizado para atualizar um registro existente;
- *Método DELETE*: utilizado para remover algum registro no *backend.*

Se o *backend* recusar a solicitação, a resposta será o código 400. No entanto, se a solicitação for aceita, a resposta será o código 200.

Na utilização de APIs em aplicações web, é comum o uso da *API REST*, que se baseia em métodos *HTTP*. Nesse modelo, o *frontend* envia uma requisição utilizando o protocolo *HTTP* para o *backend*, solicitando ações como adicionar, ler, excluir ou atualizar dados, por meio dos métodos *GET, POST, PUT e DELETE,* Da mesma forma, se o *backend* aceitar a solicitação, a resposta será o código *200*; caso contrário, a resposta será o código *400.* Existem três tipos de APIs: privadas, que são utilizadas internamente entre as aplicações de uma empresa; de parceiros, que são utilizadas entre parceiros de negócios; e públicas, que podem ser utilizadas livremente por qualquer desenvolvedor ou organização (Castro, 2022)

Segundo Noleto (2020), os *frameworks* consistem em uma série de códigos prontos que auxiliam no desenvolvimento de *software*. Eles oferecem padrões que facilitam a codificação em certas áreas, eliminando atividades repetitivas para os desenvolvedores. Embora o conceito de *framework* seja semelhante ao de biblioteca, os *frameworks* representam um conjunto maior e mais robusto de bibliotecas, permitindo configurar partes mais amplas do código.

De acordo com Bahgat (2023), alguns exemplos de *frameworks* na linguagem *Python* incluem:

- *Flask:* Um *microframework* que não requer a instalação de bibliotecas externas. Desenvolvido em 2011 por Armin Ronacher, o *Flask* já vem com diversas tecnologias e bibliotecas para o desenvolvimento de aplicativos web, oferecendo funcionalidades como validação de formulários, *upload* de arquivos e autenticação.
- *Django:* Um framework sofisticado baseado em *Python*, com recursos como *layout* de *templates,* manipulação de solicitações, *cookies*, validação de formulários e outras funcionalidades amplamente utilizadas por desenvolvedores. O *Django* é altamente escalável e pode suportar grandes volumes de tráfego, sendo ideal para redes sociais e sistemas de gerenciamento de conteúdo

2.4 SISTEMAS DE ROTEAMENTO

Sistemas de roteirização de veículos são sistemas computacionais que, por meio de algoritmos, geralmente heurísticos, conseguem obter soluções próximas das ótimas para problemas de roteirização de veículos, consumindo menos tempo e esforço em comparação aos métodos manuais tradicionais. Atualmente, esses sistemas consideram muitos tipos de restrições, como: um ou mais depósitos, janelas de tempo e vários tipos de veículos, que torna possível a obtenção de modelos muito próximos dos reais (Melo; Ferreira Filho, 2001).

A mudança mais significativa ocorreu quando os sistemas de roteirização foram implementados em ambiente computacional. Na primeira geração, o poder computacional era limitado, e os resultados nem sempre eram conhecidos imediatamente, pois dependiam de longos tempos de processamento. Além disso, esses sistemas não contavam com recursos gráficos e interativos, o que reduzia a aceitação por parte dos usuários (Bodin, 1990). Nos anos 90, o avanço tecnológico em termos computacionais, aliado às intensas pesquisas na área de pesquisa operacional, foi fundamental tanto para o desenvolvimento de novas soluções para o Problema de Roteamento de Veículos quanto para uma maior integração desses sistemas com outros setores da empresa (como compras, vendas, produção etc.). Além disso, muitos desses sistemas passaram a incorporar a tecnologia de Sistemas de Informação Geográfica (SIG), permitindo não apenas a visualização e edição das rotas, mas também a geocodificação de todos os pontos de atendimento. (Melo; Ferreira Filho, 2001).

A implantação desses sistemas pode gerar ganhos significativos em termos de atendimento e faturamento para a empresa que os adota. No entanto, tanto a aquisição e manutenção desses sistemas quanto a criação e atualização de suas bases de dados geram custos elevados, o que representa um ponto negativo e pode levar à resistência por parte dos clientes. Na prática, um sistema mal implementado e mal gerenciado pode se tornar uma fonte constante de problemas e prejuízos (Melo; Ferreira Filho, 2001).

2.4.1 SISTEMAS DE ROTEAMENTO PAGOS

No Brasil, diversos sistemas de roteirização são comercializados, cada um com características e funcionalidades específicas. Segundo informações dos sites dos fabricantes, pode-se citar como exemplo os seguintes sistemas:

- TOTVS Roteirização: Este sistema permite não apenas a roteirização dos veículos, mas também o rastreamento das cargas da empresa. É recomendado para empresas de médio e grande porte devido ao alto investimento financeiro necessário (TOTVs, 2024);
- RoutEasy: Trata-se de uma plataforma de roteirização e planejamento que busca automatizar processos manuais por meio de parametrizações. Um exemplo é o despacho automático, que é ativado ao identificar um motorista próximo à região desejada (RoutEasy, 2024);
- Cobli: É uma plataforma de gestão de frotas que, além de oferecer roteirização, permite o planejamento de rotas para até 300 endereços por importação. O usuário pode escolher entre diferentes objetivos, como reduzir a quilometragem ou agilizar as entregas, além de optar por utilizar toda a frota ou parte dela. A plataforma também disponibiliza aos clientes o acompanhamento das entregas por meio de um link de rastreamento (Cobli, 2024)
- Agile: Trata-se de um roteirizador desenvolvido pela Intelipost, que visa reduzir os custos logísticos em até 30%. A solução oferece otimização de entregas com o uso de inteligência artificial, além de incluir um aplicativo próprio para facilitar a troca de informações com os motoristas e a comprovação de entregas (Agile, 2024)

Muitos sistemas de roteamento utilizam o *Google Maps* para visualização de mapas e rotas, por meio de sua *API*. Embora exista uma versão gratuita, ela é limitada em termos de uso. O *Google Maps Platform* funciona com base em solicitações de informações à sua *API*, e os custos variam de acordo com a quantidade de solicitações feitas pelo usuário (Google, 2024).

2.4.2 SISTEMAS DE ROTEAMENTO OPEN SOURCE

Os *softwares open source* são programas livres desenvolvidos e mantidos de forma colaborativa, com o código-fonte aberto para que qualquer pessoa possa

utilizá-lo, investigá-lo, modificá-lo e redistribuí-lo. O princípio desses *softwares* é beneficiar iniciativas que não buscam lucro no seu uso. Esses projetos são geralmente centralizados em repositórios que facilitam o controle de versão, a gestão de atividades e as alterações feitas pela comunidade. Os repositórios mais utilizados incluem *GitHub, Bitbucket, SourceForge* e *Google Code* (IBM, 2024).

Segundo a IBM (2024), essa iniciativa vem ganhando grande escalabilidade ao longo dos anos. Por exemplo, o *GitHub* registrou 83 milhões de desenvolvedores e mais de 20 milhões de projetos abertos. Muitas organizações sem fins lucrativos passaram a apoiar e financiar a manutenção contínua de projetos de *software* livre, como a *Free Software Foundation* e a *Open Source Initiative* (OSI), além de várias fundações específicas de aplicativos.

O *VRPy* é uma biblioteca Python voltada para a resolução do problema de roteamento de veículos, utilizando a abordagem de geração de colunas. Nesse processo, as rotas (colunas) são geradas por meio de um problema de precificação e, em seguida, alimentadas para um problema mestre que seleciona as melhores rotas, de modo que cada vértice seja atendido exatamente uma vez. O *VRPy* não garante necessariamente uma solução ideal; para isso, o procedimento de geração de colunas deve ser inserido em um esquema de ramificação e associação (Montagné; Sanchez, 2020).

O *OR-Tools* é um software *open source* desenvolvido e mantido pela Google desde 2015, voltado para a resolução de problemas de otimização, como o problema de roteamento de veículos. Ele permite o uso de outros *solvers* (solucionador), além do Google, como o *Gurobi*. O *OR-Tools* pode ser utilizado em diversas linguagens de programação, incluindo *Python, Java, C*++ e *C#*. (OR-Tools, 2024).

O VRPSolverEasy é uma biblioteca *Python* voltada para a resolução de variantes do Problema de Roteamento de Veículos (PRV), utilizando algoritmos sofisticados que oferecem excelente desempenho. Ele foi desenvolvido para facilitar o uso entre profissionais de roteamento, eliminando a necessidade de um entendimento profundo do complexo modelo matemático que o VRPSolver oferece. Com uma interface simplificada em *Python*, o usuário pode realizar o roteamento utilizando termos predefinidos, como depósitos, clientes e tipos de veículos, que são preenchidos de maneira prática pelo próprio usuário (Errami *et al.*, 2024)

O *OpenStreetMap* (OSM) é um projeto colaborativo que cria e fornece dados para aplicações web baseadas em mapas. Ele funciona por meio de contribuições de pessoas que atualizam a plataforma, fornecendo informações sobre suas localidades, sendo um projeto colaborativo e gratuito. Inspirado nos princípios da Wikipedia, o *OSM* foi desenvolvido em 2004 por Steve Coast, estudante de computação da University College London (UCL). Inicialmente, o mapeamento seria feito apenas no Reino Unido, mas a ferramenta rapidamente despertou interesse global (Ramm, 2010).

Um dos grandes desafios de ferramentas como o *OSM*, que dependem da colaboração dos usuários, é justamente essa dependência da participação ativa (Nielsen, 2006), o que pode resultar em áreas específicas que ainda não estejam mapeadas. Em 2012, como forma de protesto contra os altos preços do Google Maps, a Apple lançou sua própria plataforma de mapeamento, utilizando dados do software *TomTom* juntamente com os do *OSM* (Medeiros, 2017).

De acordo com o site oficial da ferramenta, o *OSM* já ultrapassa a marca de 10 milhões de usuários registrados. Seu crescimento foi acelerado: em 2010, a plataforma contava com apenas 20 mil usuários; em 2015, esse número saltou para 2 milhões, e os dados mais recentes apontam que o *OSM* agora conta com 10 milhões de usuários em todo o mundo (OSM, 2024).

O *Open Source Routing Machine* (*OSRM*) segundo Rodrigues (2024) é um servidor de roteamento de alto desempenho, desenvolvido em *C++,* que utiliza o algoritmo de hierarquias de contração ou o algoritmo de Dijkstra de vários níveis para calcular o caminho mais curto. Além disso, ele se baseia nos dados do *OpenStreetMap* para realizar os cálculos das rotas mais curtas.

3 PROCEDIMENTOS METODOLÓGICOS

Este tópico aborda a metodologia empregada no desenvolvimento de uma aplicação web para a resolução de PRVs. A metodologia inclui a classificação da pesquisa e as etapas que levaram ao resultado.

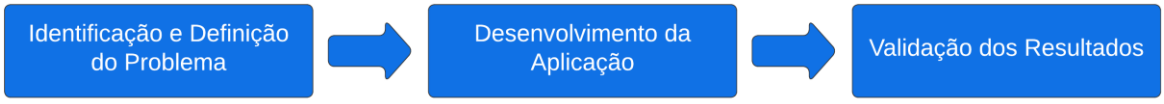
3.1 CLASSIFICAÇÃO DA PESQUISA

Esta pesquisa pode ser classificada como pesquisa aplicada, pois ela concentra-se em torno de problemas presentes nas atividades das instituições e organizações e se empenha na elaboração de busca de soluções (Thiollent, 2022). A pesquisa também pode ser classificada também por ser quantitativa, pois segundo Silva e Simon (2005) existe um problema definido, com informações e a teoria sobre o objeto de conhecimento.

3.2 ETAPAS DA PESQUISA

As etapas desenvolvidas na metodologia que contribuíram para alcançar o resultado serão descritas na Figura 8.

Figura 8 - Etapas da Pesquisa



Fonte: O autor (2024).

3.2.1 IDENTIFICAÇÃO E DEFINIÇÃO DO PROBLEMA

O PRV é um problema logístico que possui grande relevância para empresas que necessitam atender a um conjunto disperso de clientes de forma eficiente, minimizando os custos operacionais. No entanto, para fins de aplicação neste trabalho, o foco será na variante do PRVJT, que incorpora restrições de janelas de

tempo nas rotas, exigindo que os veículos atendam os clientes dentro de intervalos de tempo pré-estabelecidos (Zhang et al., 2021).

A velocidade de processamento e a capacidade dos computadores crescem de forma exponencial ao passar dos anos, permitindo resolver instância do PRV cada vez maiores, o que estimula a criação de softwares para resolver as variantes do PRV (Braekers; Ramaekers; Van; Nieuwenhuyse, 2016). Atualmente, muitas empresas utilizam softwares de PRV, existe uma grande pressão para os desenvolvedores e fornecedores desenvolverem roteadores de veículo cada vez mais rápido e preciso, criando assim uma competição no mercado para o desenvolvimento de tal ferramenta, pois os clientes demandam cada vez mais o seu produto com o menor tempo possível (Partyka, 2014). Esses softwares apresentam um alto custo de aquisição e manutenção, o que torna inacessíveis para pequenas e médias empresas (Melo; Ferreira Filho, 2001). Além disso, muitos desses sistemas oferecem um pacote genérico padrão que não possui flexibilidade, o que limita a sua aplicabilidade em contextos diferenciados.

Diante dessa lacuna, foi identificado a oportunidade do desenvolvimento de uma aplicação gráfica acessível e flexível para a resolução de problemas de roteamento, afins de explicação prática o trabalho focará na variante que adiciona janelas de tempo aos clientes visitados.

3.2.2 DESENVOLVIMENTO DA APLICAÇÃO

O desenvolvimento da aplicação foi realizado em um computador local, garantindo maior controle sobre o ambiente de desenvolvimento e facilitando testes contínuos durante o processo de criação.

3.2.3 ARQUITETURA DA APLICAÇÃO

Adotou-se uma arquitetura de microsserviços, que separa o *frontend* e o *backend*, promovendo modularidade, escalabilidade e facilidade de manutenção. Esta escolha foi fundamentada nas vantagens de flexibilidade e independência funcional oferecidas pelos microsserviços, conforme descrito por Kratzke (2017) e Tapia et al. (2020).

3.2.4 TECNOLOGIA UTILIZADAS NO DESENVOLVIMENTO DA APLICAÇÃO

No *frontend*, que é a parte que o usuário vai interagir com a aplicação, foram utilizadas as seguintes tecnologias:

- *HTML, CSS* e *JavaScript*: Utilizados para a construção da interface do usuário, proporcionando uma experiência interativa e responsiva.
- *Leaflet.js*: Escolhido para a visualização de mapas devido à sua leveza, flexibilidade e integração eficiente com *APIs* de mapas como o *OpenStreetMap*.

O *Leaflet.js* e *OpenStretMap* foram preferidas em relação a outras bibliotecas de mapas como *Google Maps API* devido às suas naturezas *open source*, custo zero e vasta comunidade de suporte.

Na parte do sistema dedicada ao servidor, foram utilizadas as seguintes tecnologias para o desenvolvimento do *backend*:

- *Python* com *Flask*: Selecionado pelo Flask ser um microframework leve e flexível, adequado para a criação métodos *HTTP* necessários para a comunicação entre *frontend* e *backend.*

O *Flask* foi escolhido em detrimento de frameworks mais robustos como *Django*, devido à sua simplicidade e facilidade de integração com outras bibliotecas. Além das tecnologias utilizadas acima, também foram utilizadas as seguintes tecnologias adicionais:

- *Pandas*: Para manipulação e processamento de dados provenientes das planilhas eletrônicas;
- *Requests:* Para realizar requisições às *APIs* externas utilizadas no projeto;
- *PuLP:* Biblioteca de programação linear empregada para modelar e resolver o PRVJT;
- *CBC_CMD*: Solucionador utilizado para resolver os modelos de programação linear;
- *Open Source Routing Machine*: Utilizado para o cálculo de distâncias e determinação das rotas mais curtas.

A aplicação foi desenvolvida e testada localmente no computador do autor, utilizando as seguintes especificações:

Hardware:

- Processador: AMD Ryzen 5 3500U (2,10 GHz)
- Memória RAM: 12 GB DDR4
- Placa de Vídeo: Radeon Vega Mobile Gfx

Software:

- Sistema Operacional: Windows 11
- Ambiente de Desenvolvimento: Visual Studio Code
- Gerenciamento de Dependências: pip (Python Package Installer)
- Controle de Versão: Git, com repositório hospedado no GitHub para facilitar a colaboração e versionamento.

3.2.5 PROCESSO DE DESENVOLVIMENTO

Inicialmente, foi realizada a configuração do ambiente, que envolveu a instalação das ferramentas necessárias, como *Python*, *Flask* e as bibliotecas *Python* pertinentes ao projeto. Em seguida, foi configurado um ambiente virtual para isolar as dependências do projeto, assegurando que as bibliotecas instaladas não interferissem com outras aplicações no sistema.

O desenvolvimento do *frontend* começou com a criação da interface gráfica utilizando *HTML, CSS e JavaScript*, proporcionando uma experiência interativa e responsiva para o usuário. A integração da biblioteca *Leaflet.js* foi essencial para a exibição e interação com mapas, permitindo a visualização geográfica das rotas. Além disso, foram implementados dois modos de operação na interface: o modo interativo, que permite a inserção manual dos dados através de cliques no mapa e preenchimento de formulários, e o modo de upload de planilhas, que facilita a inserção automatizada de dados por meio de arquivos .xlsx.

Paralelamente, foi desenvolvido o *backend* utilizando *Python* e o *framework Flask*. A configuração do servidor *Flask* permitiu gerenciar as requisições *HTTP* entre o *frontend* e o *backend* de maneira eficiente. Foram implementadas rotas específicas para o processamento dos dados dos clientes e veículos, bem como para a *geocodificação* utilizando a *API Nominatim* do *OpenStreetMap*. A resolução do modelo matemático do PRVJT foi realizada com a ajuda da biblioteca *PuLP*, que facilitou a modelagem e solução do problema de otimização. A integração com o

Open Source Routing Machine foi fundamental para o cálculo das distâncias e a geração das rotas otimizadas, garantindo que as soluções apresentadas fossem eficientes e viáveis.

O desenvolvimento foi conduzido no ambiente local do pesquisador, utilizando um computador equipado com um processador AMD Ryzen 5 3500U (2,10 GHz), 12 GB de RAM DDR4 e uma placa de vídeo integrada Radeon Vega Mobile Gfx. O sistema operacional utilizado foi o Windows 11, e o ambiente de desenvolvimento foi gerenciado com o *Visual Studio Code*. O controle de versão foi realizado utilizando *Git*, com o repositório hospedado no *GitHub* para facilitar o versionamento contínuo do código.

3.2.6 VALIDAÇÃO DOS RESULTADOS

A validação da aplicação foi conduzida através da geração e resolução de sete instâncias distintas do PRVJT, localizadas em diferentes cidades. As etapas da validação incluíram:

1. **Criação de Instâncias:**
    - Definição de demandas e janelas de tempo para cada cliente, conforme detalhado na Tabela 3.
2. **Inserção de Dados:**
    - Configuração de dois veículos com capacidades definidas, horários de saída do depósito e velocidades médias.
3. **Geração das Rotas:**
    - Utilização da aplicação para gerar as rotas otimizadas com base nos dados inseridos.
    - Registro do tempo de execução, valor da função objetivo (custo total) e número de iterações realizadas pelo solver.
4. **Análise dos Resultados:**
    - Verificação da conformidade das rotas geradas com as janelas de tempo e capacidades dos veículos.

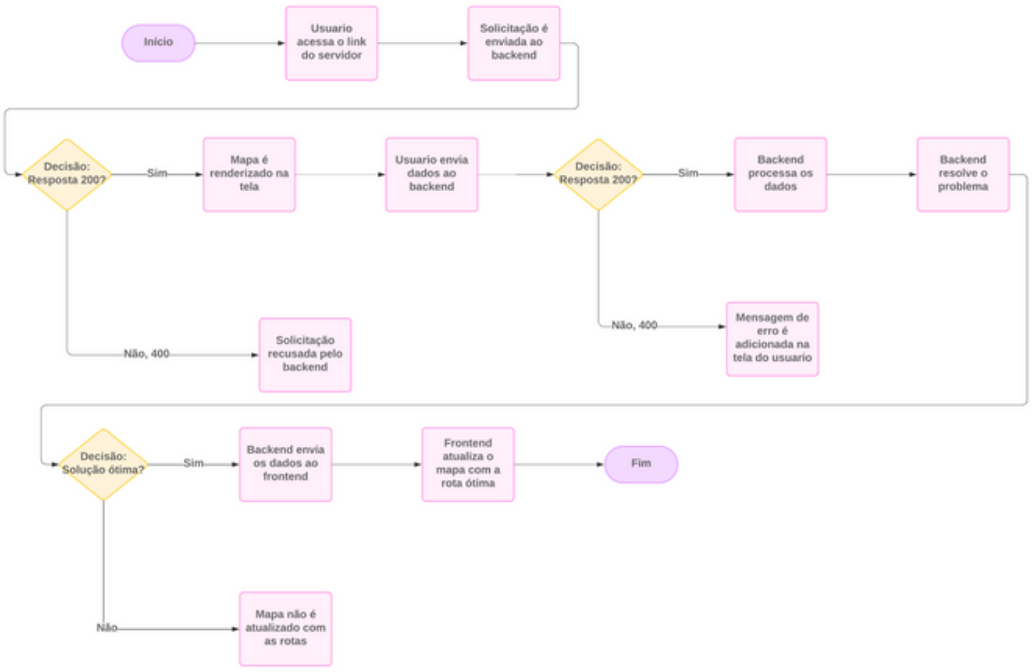
As métricas de avaliação utilizadas na validação dos resultados incluem o tempo de execução, que corresponde ao tempo necessário para resolver as

instâncias do PRVJT; o valor da função objetivo, representando o custo total das rotas geradas em termos de distância percorrida; o número de iterações realizadas pelo solver para alcançar a solução; e a conformidade com as restrições, que verifica se todas as janelas de tempo e capacidades dos veículos foram respeitadas.

4 APLICAÇÃO GRÁFICA PARA RESOLUÇÃO DE PROBLEMAS DE ROTEAMENTO DE VEÍCULOS

Neste capítulo, será apresentado o processo de desenvolvimento da aplicação gráfica para resolução de problemas de veículos. De forma específica, no trabalho em questão, considerou-se o PRVJT. No entanto, qualquer outra variante de PRV poderia ter sido considerada na aplicação. Para resolução do PRVJT, considerou-se a formulação clássica de 3 índices apresentada na Seção 2. A Figura 9, resume o que a aplicação web faz, no geral, trabalhando com requisições entre o *backend* e o *frontend,* que serão detalhados logo a frente.

Figura 9 - Fluxograma Geral da Aplicação



Fonte: O autor (2024).

4.1 FRONTEND

É a arquitetura da aplicação responsável pela parte gráfica e pela interação entre o usuário e o servidor. Ela foi dividida em dois modos de uso: o modo

interativo, que permite a interação direta do usuário com o mapa, e o modo de upload de planilha, que facilita o envio de dados por meio de planilha.

Para a exibição do mapa ao usuário, foi utilizada a biblioteca *Leaflet.js*, A Figura 10 demostra como é essa visualização. O *Leaflet* é carregado no *HTML*, e o estilo do mapa é configurado, incluindo o tamanho e as margens. Posteriormente, no *JavaScript*, é criada uma instância do mapa e definida uma visualização inicial. E então, são configuradas coordenadas de latitude e longitude iniciais, no contexto da aplicação, as coordenadas de João Pessoa foram definidas como padrão. Em seguida, é criada a camada de *tiles*, que é utilizada ao renderizar o mapa. A localização do usuário é então capturada utilizando a *API* interna do navegador, obtendo os dados de latitude e longitude, o que permite atualizar a visualização do mapa.

Figura 10 - Mapa interativo utilizando OSM e Leaflet.js

Fonte: O autor (2024).

4.1.1 MODO INTERATIVO

É o modo que adiciona informações de forma manual pelo usuário, através de cliques no mapa e é configurado por meio de um evento em *JavaScript*. Primeiramente, verifica-se se o depósito já foi adicionado. Caso não tenha sido, ao clicar no mapa, o usuário vê um *pop-up* com um formulário *HTML* que captura a latitude e longitude do depósito, além de solicitar que o usuário preencha informações como o horário de chegada ao depósito (Figura 11). Se o depósito já foi

adicionado, outro *pop-up* é exibido na tela do usuário quando ele clica no mapa, com um formulário para adicionar um cliente, contendo informações como:

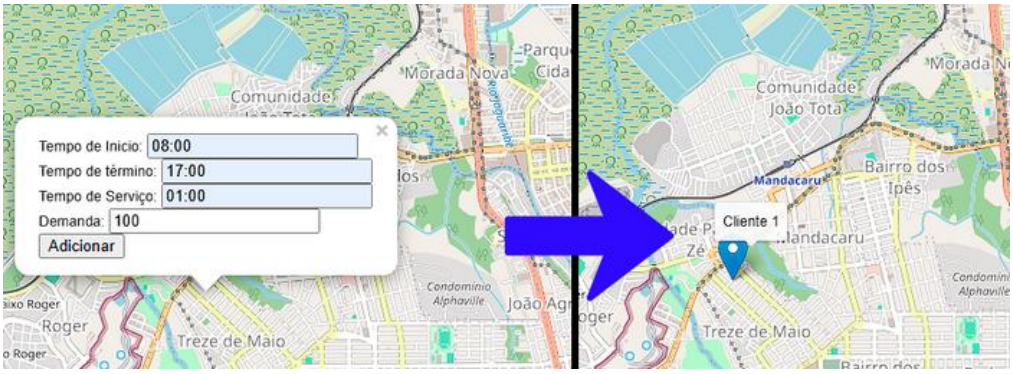
- ***Tempo de início:*** Hora de chegada
- ***Tempo de término***: Hora limite de chegada
- ***Tempo de serviço***: Tempo de serviço
- ***Demanda***: demanda do cliente

Figura 11 - Definindo o galpão

Fonte: O autor (2024).

Na Figura 11, é demonstrado como é feita a definição do galpão, no exemplo da ilustração, a hora de chegada ao galpão é definida para as 18h, quando o usuário clica em definir o galpão, um ícone ilustrativo é exibido no mapa. Depois de definir o galpão, ao clicar no mapa novamente, será exibido um *pop-up* com as informações do primeiro cliente, como mostrado na Figura 12. Nesse momento, o usuário informa o tempo de início, o tempo de término, o tempo de serviço e a demanda. Em seguida, clica em "Adicionar". Após essa ação, um ícone do cliente é exibido no mapa, com uma *tag* indicando a sua numeração. Após a configuração inicial, essas informações são enviadas ao *backend* quando o *frontend* faz uma solicitação. A Figura 13 ilustra o depósito adicionado junto com os seus clientes. Quando o usuário clica em “Enviar Clientes”, uma requisição é feita ao *backend* para armazenar essas informações.

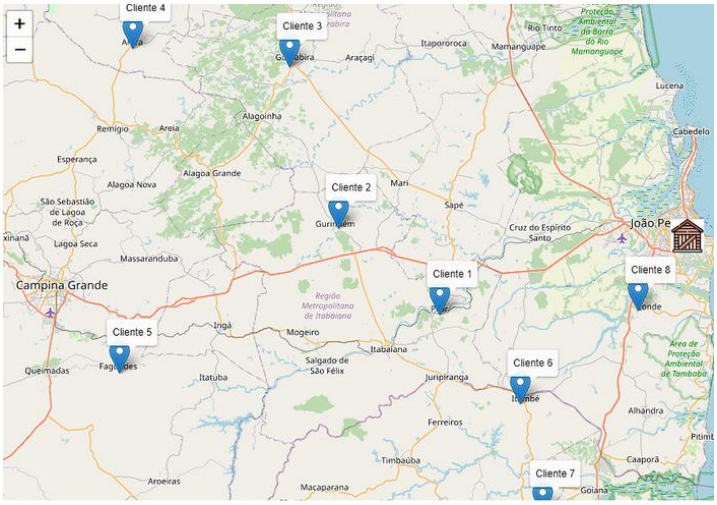
Figura 12 - Definindo Clientes



Fonte: O autor (2024).

Após serem recebidos pelo servidor, os dados são validados e uma resposta é enviada para o *frontend*. Se os dados foram preenchidos corretamente no formato padrão, a resposta será 200; caso contrário, a resposta será 400, indicando que o servidor recusou os dados enviados. Além das informações de depósito e cliente, também é possível enviar as informações dos veículos, como quantidade de veículos, capacidade de cada veículo, horário de saída de cada um e velocidade média.

Figura 13 - Clientes e Depósitos definidos

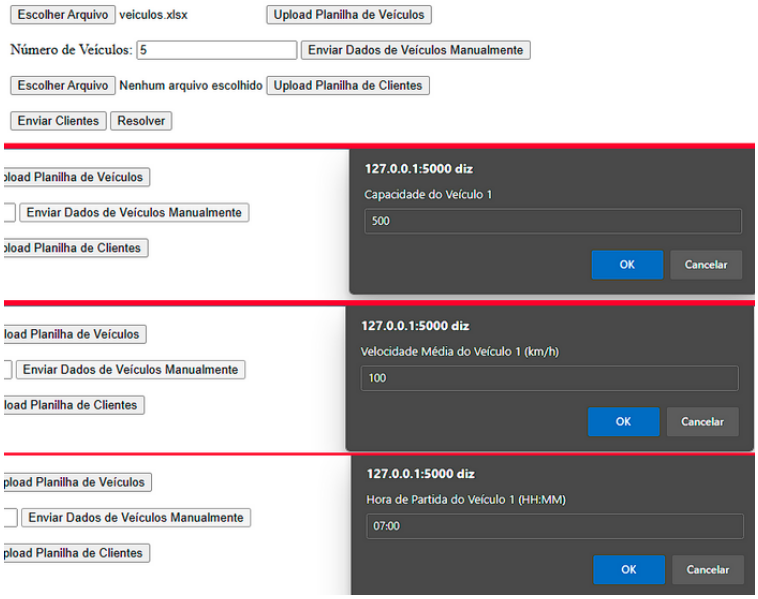


Fonte: O autor (2024).

A Figura 14 ilustra a definição dos veículos, no exemplo foram definidos cinco veículos, que estão na caixa de texto "Número de Veículos", após isso o usuário clica em "enviar dados de Veículos Manualmente", logo após aparece um *pop-up* para ser preenchido com capacidade, velocidade Média e horário de saída por

veículo. Por fim, o usuário clica em “Resolver” e uma solicitação é enviada ao *backend,* que aciona a função de resolução do problema, se tudo ocorrer bem, o mapa é atualizado com as rotas e a tabela com os tempos de viagem e uma resposta *200* é enviada ao *frontend*, caso contrário, se acontecer algum problema, que provavelmente alguma restrição foi violada ou a solução é inviável, uma resposta *400* é enviada.

Figura 14 - Definindo os dados dos veículos

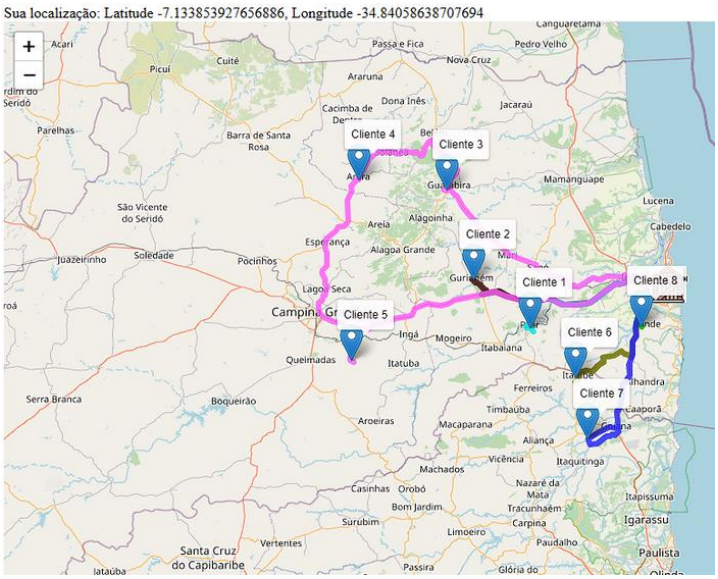


Fonte: O autor (2024).

No exemplo da Figura 15, todos os veículos tiveram capacidades iguais a 500 volumes e horários de saída as 07h, com velocidades médias definidas a 100 km/h. O horário de chegada ao deposito foi estabelecido um limite de até as 22h. Para os clientes, as janelas de tempo foram:

- **Tempo de Início**: 08h
- **Tempo de Serviço**: 30 min
- **Tempo de Término**: 18h
- **Demanda**: 100

Figura 15 - Mapa atualizado com as rotas



Fonte: O autor (2024).

Além de mostrar o mapa atualizado, o aplicativo também exibe tabelas por veículo, informando o endereço, a chegada ao cliente, tempo de serviço e horário de saída de cada cliente. A figura 16, ilustra os detalhes da rota para o veículo 5, onde ele sai do galpão para o cliente 3 e chega as 09h11min e saí as 09h41min, depois ele vai para o cliente 4, depois para o cliente 5 até retornar ao deposito as 13h18min.

Figura 16 - Tabela com as rotas

Veiculo 5

De	Para	Endereço	Chegada	Serviço	Saída
0	3	Guarabira, Região Geográfica Imediata de Guarabira, Região Metropolitana de Guarabira, Região Geográfica Intermediária de João Pessoa, Paraíba, Região Nordeste, 58200-000, Brasil	09:11	00:30	09:41
3	4	Arara, Região Geográfica Imediata de Guarabira, Região Geográfica Intermediária de João Pessoa, Paraíba, Região Nordeste, Brasil	10:16	00:30	10:46
4	5	Apertar na Hora, Fagundes, Região Geográfica Imediata de Campina Grande, Região Metropolitana de Campina Grande, Região Geográfica Intermediária de Campina Grande, Paraíba, Região Nordeste, 58487-000, Brasil	11:33	00:30	12:03
5	0	Rua Bancário Waldemar de Mesquita Accioly, Bancários, João Pessoa, Região Geográfica Imediata de João Pessoa, Região Metropolitana de João Pessoa, Região Geográfica Intermediária de João Pessoa, Paraíba, Região Nordeste, 58051-400, Brasil	13:18	00:00	13:18

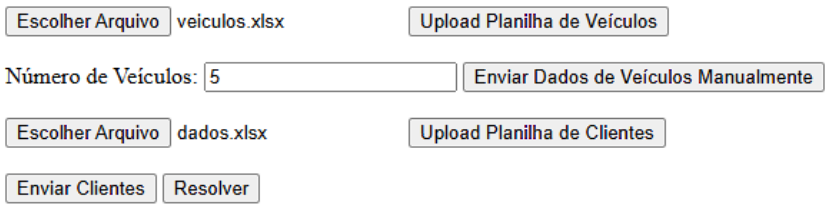
Fonte: O autor (2024).

4.1.2 MODO UPLOAD DE PLANILHA

Além da adição manual das informações, também existe a possibilidade de enviá-las por meio de uma planilha eletrônica no formato .xlsx, o que operacionalmente é mais eficiente e automatiza o processo de coleta de dados. O usuário clica em “Escolher Arquivo” (Figura 17), e um *pop-up* aparece na tela para que ele selecione a planilha com os dados. O primeiro botão refere-se à planilha

contendo as informações dos veículos, enquanto o segundo botão é destinado aos dados dos clientes.

Figura 17 - Menu do aplicativo



Fonte: O autor (2024).

Quando o usuário seleciona as planilhas, ele deve clicar em “Upload Planilha de Veículos” e “Upload Planilha de Clientes” para enviar os dados para o *backend.* Esses dados são então geocodificados, encontrando as latitudes e longitudes correspondentes, utiliza-se a *API Nominatim* do *Openstreetmap* para esse procedimento.

Tabela 1 - Tabela com os dados dos Clientes

CEP	Ready Time	Due Date	Service Time	Demand
58015-020	00:00	22:30	00:00	0,00
58036-002	08:00	17:00	00:15	100,00
58356-000	08:00	15:00	00:15	100,00
58700-010	08:00	18:00	00:15	200,00
58900-000	08:00	20:00	00:30	300,00
58910-000	08:00	20:00	00:15	100,00
55920-000	08:00	18:00	00:15	100,00
58275-000	08:00	15:00	00:15	100,00
58489-000	08:00	15:00	00:30	100,00
58360-000	08:00	15:00	00:30	100,00
55660-000	08:00	18:00	00:30	100,00
59663-000	08:00	18:00	00:45	200,00

Fonte: O autor (2024).

Nos exemplos a seguir, foram utilizados os dados da Tabela 01, que servem como padrão para garantir que não ocorram erros no servidor. Da mesma forma, existe um padrão específico a ser seguido para o envio da planilha de veículos, conforme destacado na Tabela 2. Nessa tabela, observa-se que as capacidades, velocidades e horários de saída devem ser preenchidos nas colunas, variando conforme o número do veículo, como exemplificado. A Figura 18 ilustra os clientes que foram adicionados após o *backend* retornar com suas respectivas localizações.

Figura 18 - Mapa atualizado com os clientes

Fonte: O autor (2024).

Após isso, o usuário pode clicar em 'Resolver'. A partir daí, os dados são enviados para o *backend*, em especial para a rota responsável pela resolução do modelo. Além da rota no mapa, a solução também fornece a sequência de clientes a serem visitados e uma estimativa dos horários de chegada e saída.

Tabela 2 - Dados dos Veículos

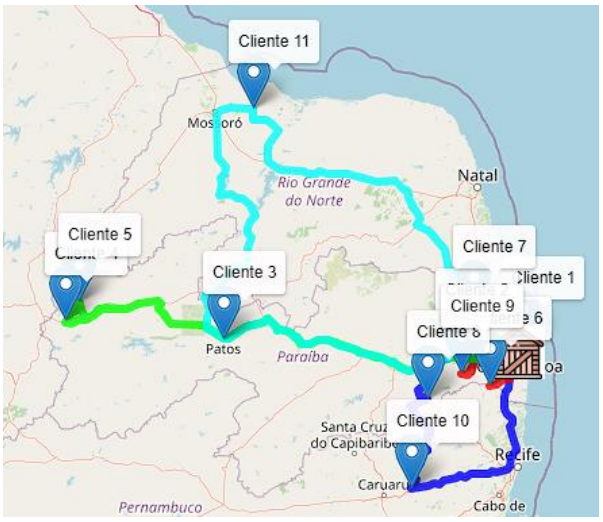
Quantidade	Capacidade V01	Velocidade V01	Horário de Saída V01
4	500	100	07:00

Fonte: O autor (2024).

A Figura 19 mostra um exemplo de resolução, no qual as rotas foram traçadas para cada veículo, que se destacam pelas cores diferentes. Da mesma

forma que a resolução pelo método manual, também é mostrado ao usuário uma tabela com os detalhes das rotas

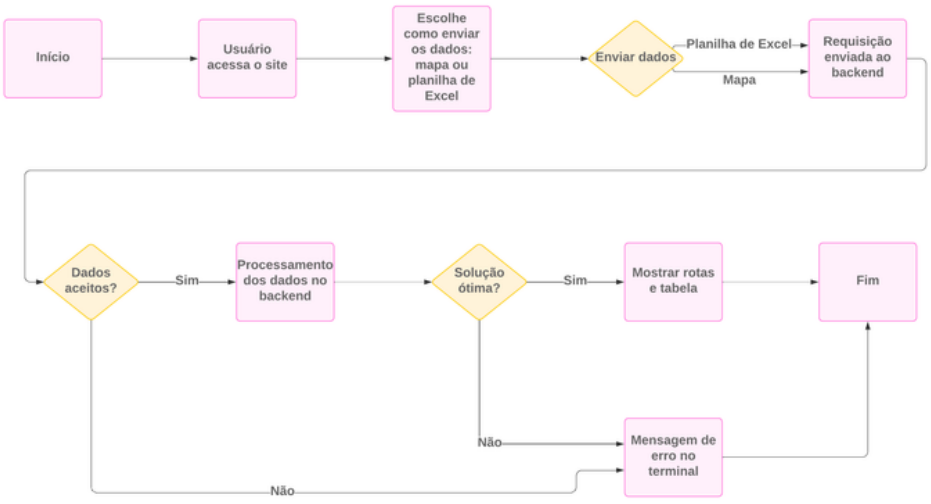
Figura 19 - Mapa atualizado com as rotas



Fonte: O autor (2024).

O fluxograma da Figura 20, resume o que foi abordado nessa seção do capítulo. O usuário ao acessar o site, ele decide se vai enviar os dados manualmente por meio de cliques no mapa ou enviar através de planilha eletrônica, então esses dados são enviados para o servidor e processados, no fim o usuário recebe as rotas por veículo e tabela com as viagens detalhadas.

Figura 20 - Fluxograma Frontend



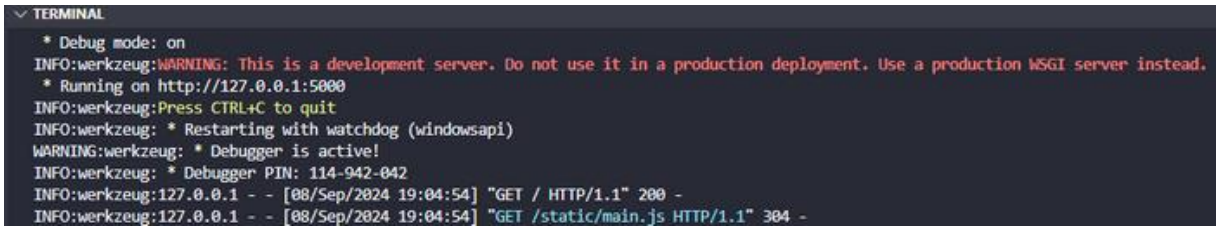
Fonte: O autor (2024).

4.2 BACKEND

Os dados podem ser fornecidos de duas maneiras: através de cliques no mapa com preenchimento de pop-up ou por upload de planilhas eletrônicas, como descrito na seção 4.1. No *backend*, quando o usuário clica no mapa, as informações de latitude e longitude são capturadas, além dos dados inseridos no formulário, como demanda e janelas de tempo. O *frontend*, ao enviar essas informações para o *backend*, aciona uma função que utiliza a *API* externa do *OSRM* para calcular as distâncias entre os clientes, determinando as distâncias entre dois pontos. No contexto do projeto, a *API* também seleciona o melhor caminho entre possíveis para todos os clientes e passa essa rota otimizada para o modelo. As informações dos veículos são enviadas por meio de preenchimento de formulários, onde são inseridos dados específicos para cada veículo.

Entre o *frontend* (cliente) e o *backend* (servidor), são utilizados métodos *HTTP* para a comunicação. O método *GET* é usado para recuperar informações do servidor, sendo empregado na aplicação para renderizar a tela inicial. O método *POST* é utilizado para enviar dados ao servidor e modificar informações no *backend*. Ele é amplamente utilizado em várias rotas do *Flask*, como para o *upload* de planilhas e o processamento de dados JSON, configurando clientes, veículos e o depósito, incluindo demanda, janelas de tempo e capacidade dos veículos, a fim de atualizar as matrizes globais que alimentam o modelo matemático. Além disso, o método *POST* é utilizado na rota de resolução do modelo, pois o solver, ao resolver o problema, gera novos dados que são utilizados na interface do usuário, como as rotas e a tabela de resultados.

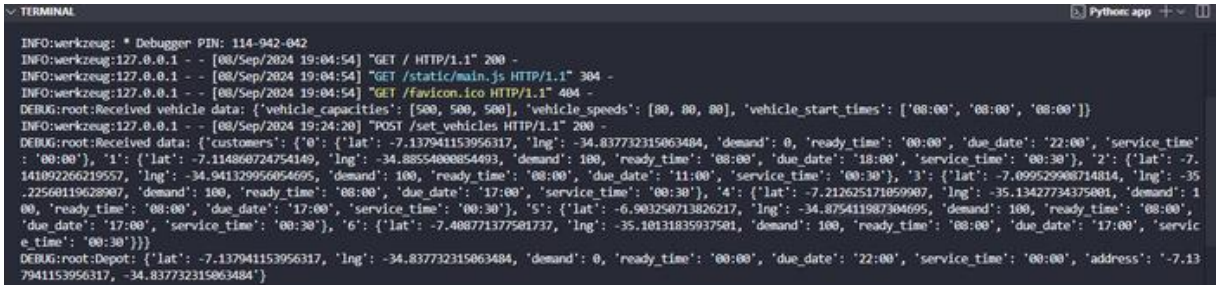
Figura 21 - Terminal quando o servidor é executado



Fonte: O autor (2024).

A Figura 21 apresenta o terminal do servidor quando o programa é iniciado, um *link* local é disponibilizado para acesso ao site, ao acessá-lo o método *GET* é acionado, renderizando o site na tela do usuário. No exemplo, a resposta do *backend* foi *200,* indicando que o site foi renderizado, enquanto o *Javascript* retornou *304*, o que significa que não teve mudança desde a última vez que foi requisitado pelo cliente.

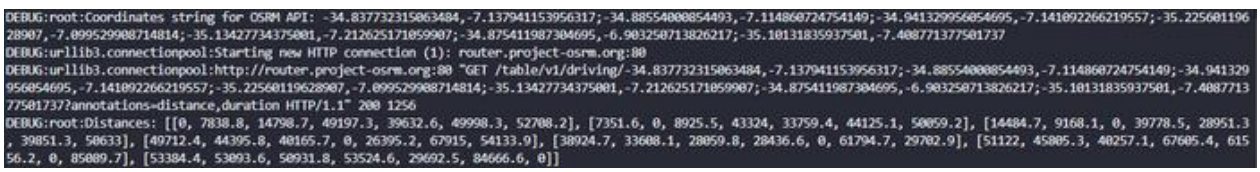
Figura 22 - Terminal com os dados dos clientes e veículos



Fonte: O autor (2024).

Quando o usuário preenche as informações e envia para o *backend,* essas informações são processadas e armazenadas em dicionários e listas *python,* como está ilustrado na Figura 22, as capacidades, velocidades e horários de saída estão sendo armazenados em listas, enquanto as informações dos clientes e deposito são armazenados em dicionários *python*. As latitudes e longitudes são passadas a *API* do *OSRM*, que calcula as distâncias entre os pontos, a *API* retorna esses dados em formato *JSON*, que são armazenados em listas *python*, a Figura 23 exemplifica essas informações, as coordenadas de latitude e longitude são passadas a *API* que retorna a distância entre os pontos. Essas informações são armazenadas na lista *distances*.

Figura 23 - Terminal com a resposta da API

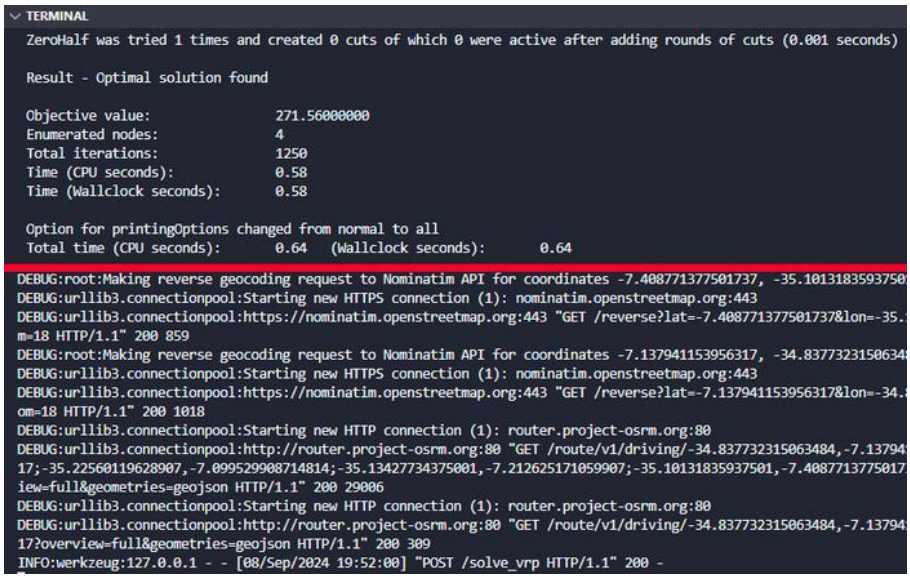


Fonte: O autor (2024).

Quando o usuário envia a solicitação ao *backend,* que deseja a solução do problema, uma rota do *Flask* de resolução é ativada, então todos esses dados são enviados para o modelo, ao resolver o modelo o valor da função objetivo é mostrado no terminal, junto com as requisições para a *API* do *OSRM* com as geometrias das rotas, e então essas informações são enviadas para o *frontend*. A Figura 24, demostra esses resultados, o valor da função objetivo para o exemplo em questão, é 271,56 Km, logo abaixo estão as requisições a *API* do *OSRM.* Além disso, o terminal também mostra uma chamada a *API* do *OSM*, que faz uma geocodificação reversa com os dados de latitude e longitude, buscando os endereços dos clientes.

A Figura 25, exemplifica como é a resposta da *API* do *OSRM*, essas informações em formato *GeoJSON*, são passadas ao *frontend,* que faz os desenhos utilizando a biblioteca *leaflet.js.* As requisições a *API* são passadas depois da definição das rotas para os veículos, os pontos fornecidos representam a sequência de locais que o veículo deve visitar.

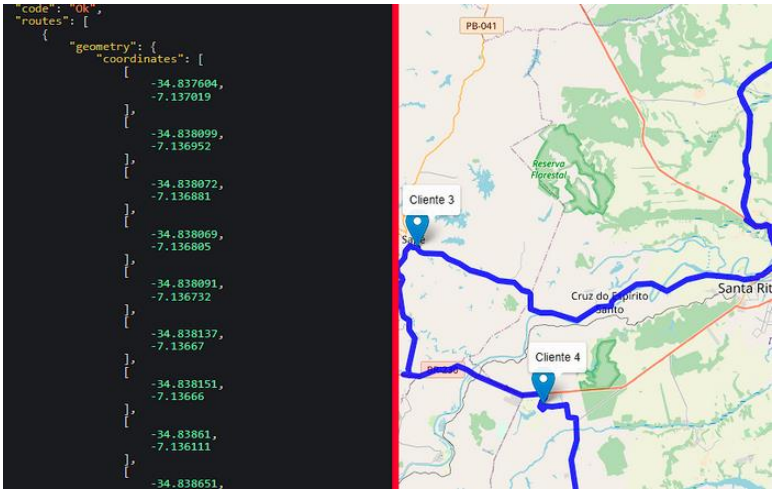
Figura 24 - Terminal com a resolução do modelo



Fonte: O autor (2024).

Também é possível enviar dados através de planilhas, como já foi explicado na seção do *frontend.* O CEP do deposito e cliente são passados para *API* do *OSM* por meio de uma solicitação *GET*, que procura na sua base de dados (latitude e longitude) as localizações correspondentes.

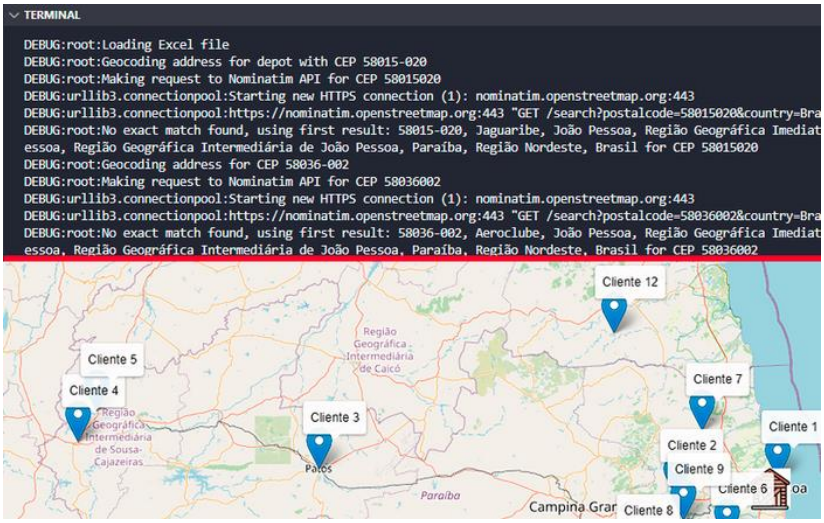
Figura 25 - Exemplificação de GeoJSon para rota no mapa



Fonte: O autor (2024).

A Figura 26 mostra como é feita a requisição entre o *backend* e a *API* do *OSM*. É feita uma requisição à *API Nominatim* do *OSM* para cada CEP fornecido. Para facilitar o processo, sempre o primeiro CEP é tratado como o depósito, e os demais são considerados os clientes. As solicitações são feitas via *HTTPS* ao servidor nominatim.openstreetmap.org

Figura 26 - Geocodificação da planilha de clientes no mapa

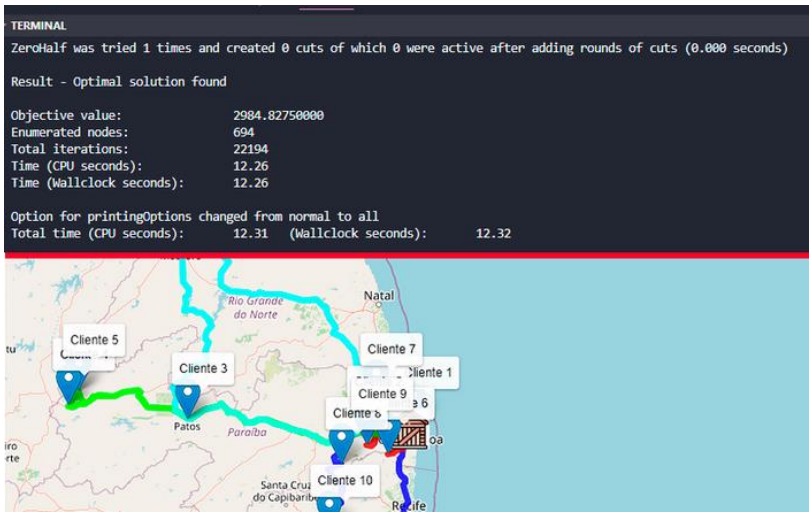


Fonte: O autor (2024).

A Figura 27 apresenta o valor da função objetivo para os dados da Tabela 01, sendo igual a 2984,8275 km, com 22.194 iterações e um tempo de resolução de 12,26 segundos. Logo abaixo, encontra-se o desenho da rota correspondente. O

restante do procedimento segue o mesmo processo realizado quando os dados são inseridos manualmente, sem qualquer alteração na lógica de cálculo ou na exibição das rotas no mapa.

Figura 27 - Rotas e valor da função objetivo



Fonte: O autor (2024).

4.3 VALIDAÇÃO DOS RESULTADOS

Na validação dos resultados, foram geradas 7 instâncias, cujas demandas e janelas de tempo estão detalhadas na Tabela 3. Observa-se que a coluna "cidade" é meramente informativa, sem impacto direto no funcionamento do *backend*. Além disso, foram utilizados 2 veículos com as seguintes características:

- **Capacidade**: 500 volumes;
- **Velocidade**: 100 km/h;
- **Horário de saída**: 07:00.

Tabela 3 - Dados dos Clientes

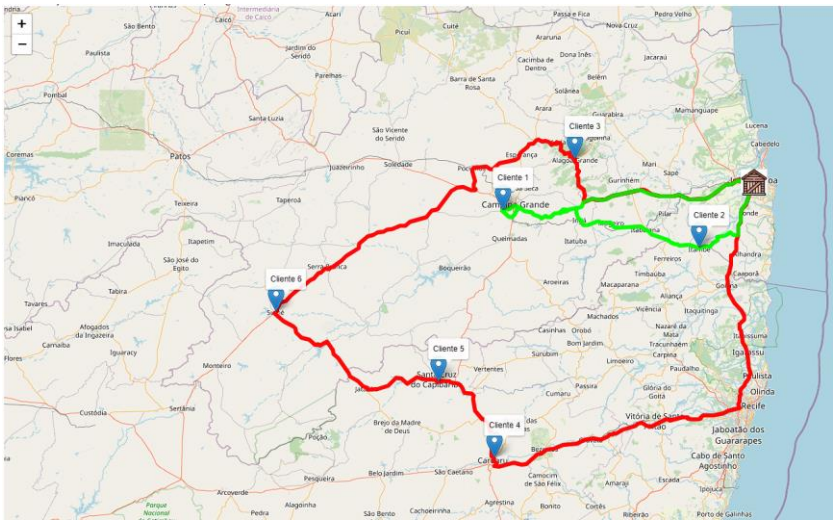
CEP	Ready Time	Due Date	Service Time	Demand	Cidade
58015-020	00:00	22:30	00:00	0,00	João pessoa
58432-570	08:00	15:00	01:00	200,00	Campina Grande
55920-000	08:00	14:00	01:00	200,00	Itambé

58388-000	10:00	17:00	00:15	50,00	Alagoa Grande
55012-085	08:00	18:00	00:15	100,00	Caruaru
55192-000	08:00	18:00	00:15	100,00	Santa Cruz do Capibaribe
58540-000	08:00	15:00	00:15	150,00	Sumé

Fonte: O autor (2024).

Com os dados da Tabela 3 carregados na aplicação, as rotas foram geradas e suas geometrias retornadas, conforme evidenciado na Figura 28. A figura mostra duas rotas distintas: uma em cor vermelha e outra em cor verde, representando a separação entre os veículos utilizados no processo de roteamento.

Figura 28 - Resultado da Validação das geometrias das rotas



Fonte: O autor (2024).

A aplicação também gera os detalhes das rotas para cada veículo. A Figura 29 apresenta as informações referentes à validação dos resultados. O veículo 0 sai do depósito às 07h, segue para o Cliente 3 (Alagoa Grande), onde chega às 10h31 e parte às 10h46. Em seguida, vai para o Cliente 6 (Sumé), chegando às 12h40 e saindo às 12h55. Depois, segue para o Cliente 5 (Santa Cruz do Capibaribe), com chegada às 13h54 e partida às 14h09. Em seguida, passa por Caruaru, chegando às 14h42 e saindo às 14h57, retornando ao depósito às 17h22.

O veículo 1, de forma semelhante, porém em uma rota distinta, sai do depósito às 07h. Ele chega no cliente 2 em Timbaúba às 08h47, partindo às 09h47. Depois, vai para o Cliente 1 (Campina Grande), chegando às 10h53 e saindo às 11h53, retornando ao depósito às 13h11.

Figura 29 - Detalhamento das Rotas (Validação)

Veículo 0

De	Para	Endereço	Chegada	Serviço	Saída
0	3	Avexado Provedor de Internet, 21, Rua Horácio de Albuquerque, Alagoa Grande, Região Geográfica Imediata de Campina Grande, Região Geográfica Intermediária de Campina Grande, Paraíba, Região Nordeste, 58388-000, Brasil	10:31	00:15	10:46
3	6	Rua Aleixo Bezerra, Sumé, Região Geográfica Imediata de Sumé, Região Geográfica Intermediária de Campina Grande, Paraíba, Região Nordeste, 58540-000, Brasil	12:40	00:15	12:55
6	5	Rua Manoel Balbino, São Cristóvão, Santa Cruz do Capibaribe, Região Geográfica Imediata de Caruaru, Região Geográfica Intermediária de Caruaru, Pernambuco, Região Nordeste, 55192-000, Brasil	13:54	00:15	14:09
5	4	Lira Hostel, 38, Rua Luiz Paes, Maurício de Nassau, Universitário, Caruaru, Região Geográfica Imediata de Caruaru, Região Geográfica Intermediária de Caruaru, Pernambuco, Região Nordeste, 55012-085, Brasil	14:42	00:15	14:57
4	0	Receita Estadual -PB, Rua das Trincheiras, Distrito Mecânico, Jaguaribe, João Pessoa, Região Geográfica Imediata de João Pessoa, Região Metropolitana de João Pessoa, Região Geográfica Intermediária de João Pessoa, Paraíba, Região Nordeste, 58015-020, Brasil	17:22	00:00	17:22

Veículo 1

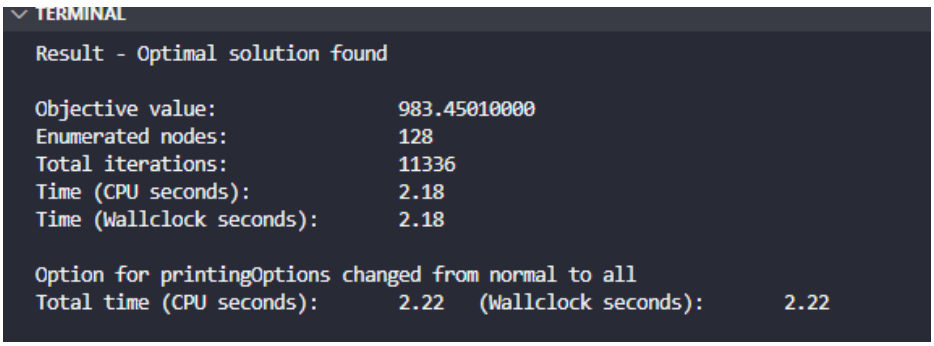
De	Para	Endereço	Chegada	Serviço	Saída
0	2	Policlínica UneSaúde, 190, Rua São Sebastião, Centro, Itambé, Região Geográfica Imediata de Goiana-Timbaúba, Região Geográfica Intermediária do Recife, Pernambuco, Região Nordeste, 55920-000, Brasil	08:47	01:00	09:47
2	1	Departamento Estadual de Transito - DETRAN, 2168, Rua Francisco Lopes de Almeida, Portal Sudoeste, Três Irmãs, Campina Grande, Região Geográfica Imediata de Campina Grande, Região Metropolitana de Campina Grande, Região Geográfica Intermediária de Campina Grande, Paraíba, Região Nordeste, 58432-570, Brasil	10:53	01:00	11:53
1	0	Receita Estadual -PB, Rua das Trincheiras, Distrito Mecânico, Jaguaribe, João Pessoa, Região Geográfica Imediata de João Pessoa, Região Metropolitana de João Pessoa, Região Geográfica Intermediária de João Pessoa, Paraíba, Região Nordeste, 58015-020, Brasil	13:11	00:00	13:11

Fonte: O autor (2024).

Com essas rotas, a aplicação demonstra a capacidade de planejar e otimizar o trajeto de cada veículo, considerando as janelas de tempo e horários de saída.

O valor da função objetivo, ilustrado na Figura 30, é de 983,45 km, obtido após 11.336 iterações, com um tempo de resolução de 2,2 segundos. Com base nesses resultados, a ferramenta demonstrou ser uma ótima solução, tanto em termos gráficos quanto de desempenho, atendendo eficientemente às necessidades do problema proposto.

Figura 30 - Valor da Função Objetivo (Validação)



Fonte: O autor (2024).

5 CONCLUSÃO

Este trabalho teve como objetivo desenvolver uma aplicação gráfica para a resolução do Problema de Roteamento de Veículos, para fins práticos o caso especial com janelas de tempo, integrando tecnologias modernas no contexto da Indústria 4.0. A aplicação proposta visa auxiliar empresas, especialmente de pequeno e médio porte, a otimizar suas operações logísticas, reduzindo custos operacionais e melhorando a eficiência no atendimento aos clientes. Através da utilização de modelos matemáticos de otimização e da integração com APIs externas, como o *OpenStreetMap* e o *Open Source Routing Machine*, foi possível criar uma ferramenta robusta e acessível. A aplicação permite a inserção de dados de forma intuitiva, seja manualmente através de uma interface gráfica interativa ou por meio do *upload* de planilhas eletrônicas, facilitando a adaptação às necessidades específicas de cada usuário.

Os resultados obtidos demonstraram a eficácia da aplicação na geração de rotas otimizadas, respeitando as restrições de capacidade dos veículos e as janelas de tempo dos clientes. Além disso, a visualização gráfica das rotas e a possibilidade de simular diferentes cenários contribuem para uma melhor tomada de decisão por parte dos gestores logísticos. Como contribuição adicional, o trabalho evidencia a viabilidade de desenvolver soluções tecnológicas acessíveis que podem ser implementadas sem a necessidade de altos investimentos financeiros, tornando-se uma alternativa viável aos sistemas de roteamento pagos, que muitas vezes não atendem plenamente às demandas específicas das empresas.

Para trabalhos futuros, sugere-se a expansão das funcionalidades da aplicação, incorporando outras variantes do PRV, como múltiplos depósitos e considerações ambientais, como a minimização de emissões de poluentes. Além disso, a implementação de algoritmos heurísticos poderia ampliar ainda mais a flexibilidade da ferramenta em gerar soluções mais rápidas. Conclui-se que a aplicação desenvolvida atende aos objetivos propostos, contribuindo significativamente para a otimização das operações logísticas e demonstrando o potencial de soluções tecnológicas alinhadas com os conceitos da Indústria 4.0.

REFERÊNCIAS

THIOLLENT, Michel. **Metodologia da pesquisa-ação**. Cortez Editora, f. 61, 2022. 122 p.

CULOT, Giovanna; NASSIMBENI, Guido; ORZES, Guido; SARTOR, Marco. Behind the definition of Industry 4.0: analysis and open questions. **International Journal Of Production Economics**, [S.L.], v. 226, p. 107617, ago. 2020. Elsevier BV. http://dx.doi.org/10.1016/j.ijpe.2020.107617

BÜCHI, Giacomo; CUGNO, Monica; CASTAGNOLI, Rebecca. Smart factory performance and Industry 4.0. **Technological Forecasting And Social Change**, [S.L.], v. 150, p. 119790, jan. 2020. Elsevier BV. http://dx.doi.org/10.1016/j.techfore.2019.119790.

ABDELMAJIED, Fathyelsayed Youssef. Industry 4.0 and Its Implications: concept, opportunities, and future directions. **Supply Chain - Recent Advances And New Perspectives In The Industry 4.0 Era**, [S.L.], v. 1, n. 1, p. 1-16, 27 jul. 2022. IntechOpen. http://dx.doi.org/10.5772/intechopen.102520

KARNIK, Niharika; BORA, Urvi; BHADRI, Karan; KADAMBI, Prasanna; DHATRAK, Pankaj. A comprehensive study on current and future trends towards the characteristics and enablers of industry 4.0. **Journal Of Industrial Information Integration**, [S.L.], v. 27, p. 100294, maio 2022. Elsevier BV. http://dx.doi.org/10.1016/j.jii.2021.100294.

PEREIRA, A.C.; ROMERO, F.. A review of the meanings and the implications of the Industry 4.0 concept. **Procedia Manufacturing**, [S.L.], v. 13, p. 1206-1214, 2017. Elsevier BV. http://dx.doi.org/10.1016/j.promfg.2017.09.032.

PETRONI, Benedito Cristiano; GLORIA JUNIOR, Irapuan; GONÇALVES, Rodrigo Franco. **SISTEMAS CIBER-FÍSICOS**. 2018. Disponível em:

https://www.researchgate.net/publication/329451552_Sistemas_Cyber_Fisicos. Acesso em: 16 set. 2024.

PAZ, Antonio Carlos Menezes; LOOS, Mauricio Johnny. A importância da computação em nuvem para a indústria 4.0. **Revista Gestão Industrial**, [S.L.], v. 16, n. 2, p. 1-20, 23 out. 2020. Universidade Tecnologica Federal do Parana (UTFPR). http://dx.doi.org/10.3895/gi.v16n2.9317

PERERA, Charith; LIU, Chi Harold; JAYAWARDENA, Srimal; CHEN, Min. **A Survey on Internet of Things From Industrial Market Perspective**. IEEE Access, [S.L.], v. 2, p. 1660-1679, 2014. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/access.2015.2389854.

GALDINO, Natanael. **Big Data: Ferramentas e Aplicabilidade**. 2016. Disponível em: https://www.aedb.br/seget/arquivos/artigos16/472427.pdf. Acesso em: 22 set. 2024.

NOVAES, Antônio Galvão. **Logística e Gerenciamento da Cadeia de Distribuição**. Rio de Janeiro: Elsevier, 2007**.**

CHRISTOPHER, Martin. **Logistics and Supply Chain Management: Strategies for Reducing Cost and Improving Service**. London: Financial Times, Prentice-Hall, 1998.

ANTZIG, G. B.; RAMSER, J. H.. The Truck Dispatching Problem. **Management Science**, [S.L.], v. 6, n. 1, p. 80-91, out. 1959. Institute for Operations Research and the Management Sciences (INFORMS). http://dx.doi.org/10.1287/mnsc.6.1.80.

ZHANG, Haifei *et al*. Review of Vehicle Routing Problems: models, classification and solving algorithms. **Archives Of Computational Methods In Engineering**, [S.L.], v. 29, n. 1, p. 195-221, 19 abr. 2021. Springer Science and Business Media LLC. http://dx.doi.org/10.1007/s11831-021-09574-x.

LENSTRA, J. K.; KAN, A. H. G. Rinnooy. Complexity of vehicle routing and scheduling problems. **Networks**, [S.L.], v. 11, n. 2, p. 221-227, jun. 1981. Wiley. http://dx.doi.org/10.1002/net.3230110211.

CUNHA, Claudio Barbieri da. **Uma contribuição para o problema de roteirização de veículos com restrições operacionais**. 1997. Tese (Doutorado em Engenharia de Transportes) - Escola Politécnica, Universidade de São Paulo, São Paulo, 1997. doi:10.11606/T.3.1997.tde-31012024-095926. Acesso em: 2024-09-29.

CUNHA, C. B. da. Aspectos práticos da aplicação de modelos de roteirização de veículos a problemas reais. **TRANSPORTES**, *[S. l.]*, v. 8, n. 2, 2000. DOI: 10.14295/transportes.v. 8i2.188. Disponível em: https://revistatransportes.org.br/anpet/article/view/188. Acesso em: 29 set. 2024.

NIELSEN, Claus Djernæs. **Investigate availability and maintainability within a microservice architecture**. 2015. Tese de Doutorado. Aarhus University.

TAPIA, Freddy; MORA, Miguel Ángel; FUERTES, Walter; AULES, Hernán; FLORES, Edwin; TOULKERIDIS, Theofilos. From Monolithic Systems to Microservices: a comparative study of performance. **Applied Sciences**, [S.L.], v. 10, n. 17, p. 5797, 21 ago. 2020. MDPI AG. http://dx.doi.org/10.3390/app10175797. Disponível em: https://www.mdpi.com/2076-3417/10/17/5797. Acesso em: 29 set. 2024.

KRATZKE, Nane. **About microservices, containers and their underestimated impact on network performance**. arXiv preprint arXiv:1710.04049, 2017.

AWS. **Microsserviços**. 2024. Disponível em: https://aws.amazon.com/pt/microservices/. Acesso em: 22 set. 2024.

ARVIND, Vidhya *et al*. **Apresentando a camada de abstração de dados de valor-chave da Netflix**. 2024. Disponível em: https://netflixtechblog.com/introducing-netflixs-key-value-data-abstraction-layer-1ea8a0a11b30. Acesso em: 29 set. 2024

MELO, André Cristiano da Silva; FERREIRA FILHO, Virgílio José Martins. SISTEMAS DE ROTEIRIZAÇÃO E PROGRAMAÇÃO DE VEÍCULOS. **Pesquisa Operacional**, [S.L.], v. 21, n. 2, p. 223-232, jul. 2001. FapUNIFESP (SciELO). http://dx.doi.org/10.1590/s0101-74382001000200007.

GOOGLE. **Encontre o produto certo para o trabalho**. 2024. Disponível em: https://mapsplatform.google.com/intl/pt-BR/products/. Acesso em: 29 set. 2024.

IBM. **O que é um software livre (open source)?** 2024. Disponível em: https://www.ibm.com/br-pt/topics/open-source. Acesso em: 29 set. 2024.

MONTAGNÉ, Romain; SANCHEZ, David Torres. **Mathematical Background**. 2020. Disponível em: https://vrpy.readthedocs.io/en/latest/mathematical_background.html. Acesso em: 29 set. 2024.

NOLETO, Cairo. **Framework: o que é, como ele funciona e para que serve?** 2020. Disponível em: https://blog.betrybe.com/framework-de-programacao/o-que-e-framework/. Acesso em: 29 set. 2024.

BAHGAT, Ahmed. **Flask vs Django: Escolhendo seu Próximo Framework Python**. 2023. Disponível em: https://kinsta.com/pt/blog/flask-vs-django. Acesso em: 29 set. 2024.

PESSOA, Artur *et al*. A generic exact solver for vehicle routing and related problems. **Mathematical Programming**, [S.L.], v. 183, n. 1-2, p. 483-523, 25 jun. 2020. Springer Science and Business Media LLC. http://dx.doi.org/10.1007/s10107-020-01523-z.

SZWARCWALD, Celia e col. **ConVid - Pesquisa de Comportamentos pela Internet durante a pandemia de COVID-19 no Brasil: concepção e metodologia de aplicação**, 2021. Disponível em: https://www.scielosp.org/article/csp/2021.v37n3/e00268320 Acesso em: 29 set. 2024

NOLETO, Cairo. **Aplicações web: entenda o que são e como funcionam!** 2022. Disponível em: https://blog.betrybe.com/desenvolvimento-web/aplicacoes-web/. Acesso em: 29 set. 2024.

ERRAMI, Najib *et al*. VRPSolverEasy: a python library for the exact solution of a rich vehicle routing problem. **Informs Journal On Computing**, [S.L.], v. 36, n. 4, p. 956-965, jul. 2024. Institute for Operations Research and the Management Sciences (INFORMS). http://dx.doi.org/10.1287/ijoc.2023.0103.

PARTYKA, Janice. **Vehicle Routing Software Survey: VR delivers the goods**. 2014. Disponível em: https://www.informs.org/ORMS-Today/Public-Articles/February-Volume-41-Number-1/Vehicle-Routing-Software-Survey-VR-delivers-the-goods. Acesso em: 29 set. 2024.

SOUZA, Ivan de. **Entenda o que é Rest API e a importância dele para o site da sua empresa**. 2020. Disponível em: https://rockcontent.com/br/blog/rest-api/. Acesso em: 29 set. 2024.

CASTRO, Thallyta. **Como o frontend e o backend se comunicam**. 2022. Disponível em: https://medium.com/@thallyta-castro-cv/como-o-frontend-e-o-backend-se-comunicam-5d41f3c5da62. Acesso em: 29 set. 2024.

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture in Pratic**. 3. ed. Massachusetts, USA: Pearson Education, Inc., 2015. ISBN 978-0-321-81573-6.

POLIDORO, Priscila. **Full stack, front-end e back-end: quais são as diferenças?** 2023. Disponível em: https://www.locaweb.com.br/blog/temas/codigo-aberto/full-stack-front-end-e-back-end-quais-sao-as-diferencas/. Acesso em: 29 set. 2024.

RODRIGUES, Diego. **OSRM - Open Source Routing Machine**. 2024. Disponível em: https://opt.com.br/osrm-open-source-routing-machine/. Acesso em: 04 out. 2024.

Ramm, F., J. Topf e S. Chilton: **OpenStreetMap: Using and Enhancing the Free Map of the World**. UIT Cambridge, 2010

NIELSEN, Jakob. **The 90-9-1 Rule for Participation Inequality in Social Media and Online Communities**. 2006. Disponível em: https://www.nngroup.com/articles/participation-inequality/. Acesso em: 04 out. 2024

MEDEIROS, Gabriel Franklin Braz de. **OpenStreetMap: Uma Análise Sobre a Evolução de Dados Geográficos Colaborativos no Brasil**. 2017. 68 f. Monografia (Especialização) - Curso de Engenharia da Computação, Ciência da Computação, Universidade de Brasília, Brasília, 2017. Disponível em: https://bdm.unb.br/bitstream/10483/19524/1/2017_GabrielFranklinBrazdeMedeiros.pdf. Acesso em: 04 out. 2024.

OSM. **Stats**. 2024. Disponível em: https://wiki.openstreetmap.org/wiki/Stats. Acesso em: 04 out. 2024.

APPLEGATE, D. L. et al. **The traveling salesman problem: a computational study**. Princeton: Princeton University Press, 2007.

TOTVS. **Roteirização e Entregas**. 2024. Disponível em: https://www.totvs.com/roteirizacao-e-entregas/. Acesso em: 04 out. 2024.

ROUTEASY. **Roteirização e gestão de entregas**. 2024. Disponível em: https://www.routeasy.com.br/solucoes/roteirizacao. Acesso em: 04 out. 2024.

COBLI. **Roteirizador para frotas**: sistema completo. 2024. Disponível em: https://www.cobli.co/roteirizador/. Acesso em: 04 out. 2024.

Ballou, Ronald H. **Business Logistics/Supply Chain Management**. 5ª edição, Prentice Hall, 1995.

LIU, Xiaobo et al. A Systematic Literature Review of Vehicle Routing Problems with Time Windows. **Sustainability**, [S.L.], v. 15, n. 15, p. 12004, 4 ago. 2023. MDPI AG. http://dx.doi.org/10.3390/su151512004. Disponível em: https://www.mdpi.com/2071-1050/15/15/12004. Acesso em: 04 out. 2024.

BODIN, L. D**. Twenty years of routing and scheduling. Operations Research**, v. 38, n. 4, p. 571-579, 1990.

SILVA, D.; SIMON, F. O**. Abordagem quantitativa de análise de dados de pesquisa: construção e validação de escala de atitude**. *Cadernos do CERU*, v. 2, n. 16, p. 11-27, 2005.

PRESA, Christian. **Redução de custo ainda é prioridade para mais da metade dos gestores de logística no Brasil**. 2023. Disponível em: https://mundologistica.com.br/noticias/reducao-de-custo-e-prioridade-na-gestao-logistica. Acesso em: 24 out. 2024.

AGILE. **MAIS AGILIDADE, MENOS CUSTOS Roteirizador de entregas e coletas.** 2024. Disponível em: https://agileprocess.com.br. Acesso em: 24 out. 2024.